Learning circulant support vector machines for fast image search

Ramin Raziperchikolaei and **Miguel Á. Carreira-Perpiñán**

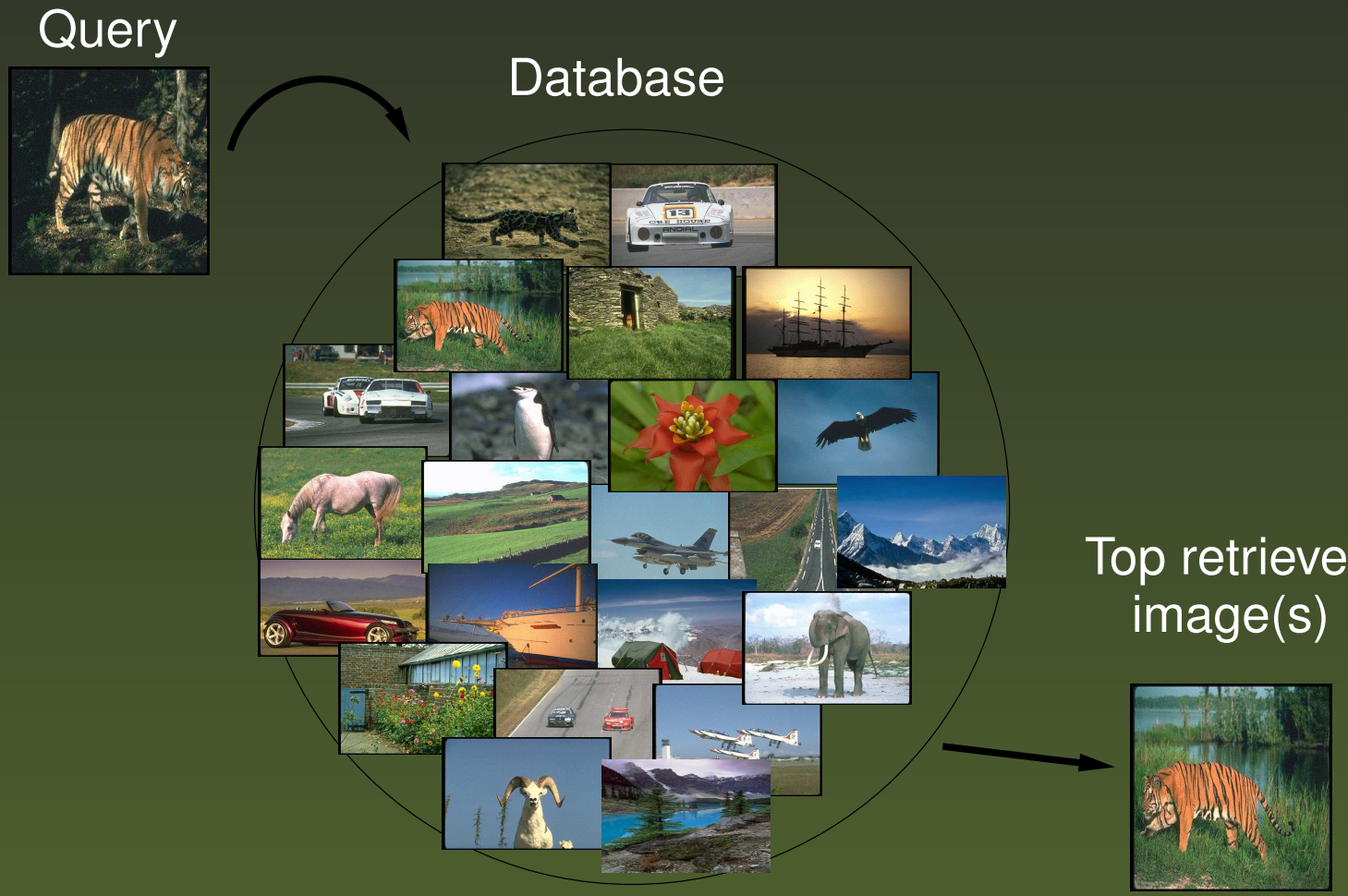
Electrical Engineering and Computer Science

University of California, Merced

http://eecs.ucmerced.edu

Large scale image retrieval

Searching a large database for images that are closest to a query.
A nearest neighbours problem on N vectors in $\mathbb{R}^D$ with large N and D.



A fast, approximate approach: binary hashing.

Large scale image retrieval: binary hash functions

A binary hash function $\mathbf{h}$ maps a high-dimensional vector $\mathbf{x} \in \mathbb{R}^D$ to a L-bit vector $\mathbf{z} = \mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})) \in \{0,1\}^L$. It should:

- preserve neighbours: map (dis)similar images to (dis)similar codes (in Hamming distance)
- be fast to compute.

image $\mathbf{x} \in \mathbb{R}^D$

binary code
$\mathbf{z} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^L$

1	1	0	1	0	0

XOR

0	1	1	1	0	1

1	0	1	0	0	1

Hamming distance = 3

Large scale image retrieval: binary hash functions

Scalability: we have millions or billions of high-dimensional images.

- Time complexity is computed based on two operations:
  - Time needed to generate the binary code for the query.
  - $\mathcal{O}(1)$ to search for similar codes using inverted index.
- Space complexity is $\mathcal{O}(NL)$. We can fit the binary codes of the entire dataset in memory, further speeding up the search.
- Time and space complexities of the exact search are both $\mathcal{O}(ND)$.

The main goal of binary hash functions is to preserve similarities:

- The similarity could be complex: very different images in the pixel space may be similar, e.g. because of a difference in viewpoint).
- So the hash function needs to be learned from a dataset with known similarities.

Approaches in learning the hash functions: { optimisation-based, diversity-based }

Learning binary hashing: optimisation-based approach

Assume we have N points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in D dimensional space: $\mathbf{x}_i \in \mathbb{R}^D$.

Consider the linear hash function $\mathbf{h}(\mathbf{x}_i) = \operatorname{sgn}(\mathbf{W}\mathbf{x}_i) \in \{-1, +1\}^L$ that maps each image into an L-bit binary code.

Optimisation-based methods define an objective $E(\mathbf{h})$ that tries to learn hash functions that map similar images into similar binary codes. They use optimisation techniques to minimise $E(\mathbf{h})$.

Examples of the objective function $E(\mathbf{h})$:

- Autoencoder (unsupervised hashing):

  encoder ($\mathbf{h}$) and decoder ($\mathbf{f}$) can be linear, neural nets, etc.

$$\min_{\mathbf{h},\mathbf{f}} E(\mathbf{h}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{h}(\mathbf{x}_n))\|^2$$

- Laplacian Loss (supervised hashing with the known similarities):

$$\min_{\mathbf{h}} E(\mathbf{h}) = \sum_{i,j=1}^{N} w_{ij} \|\mathbf{h}(\mathbf{x}_i) - \mathbf{h}(\mathbf{x}_j)\|^2 \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^{N} \mathbf{h}_l(\mathbf{x}_i) = 0 \\ \mathbf{h}(\mathbf{X})\mathbf{h}(\mathbf{X})^T = \mathbf{I} \end{cases}$$

Learning binary hashing: optimisation-based (Cont)

Many ad-hoc methods have been proposed to optimise the objectives.

A generic way to optimise the previous objectives is the Method of Auxiliary Coordinates (MAC) (Carreira-Perpiñán & Wang 2012, 2014).

This is a different algorithm that optimises the objective correctly:

1. Define the binary coordinates $\mathbf{Z} \in \{0,1\}^{N \times L}$ as the output of the hash function and minimise an equivalent constrained problem:

$$\min_{\mathbf{h},\mathbf{Z}} \mathbf{E}(\mathbf{Z}) \quad \text{s.t} \quad \mathbf{Z} = \mathbf{h}(X)$$

2. Apply the quadratic penalty method and optimise the following objective as progressively increasing μ

$$\min_{\mathbf{h},\mathbf{Z}} \mathbf{E}(\mathbf{Z}) + \mu \left\| \mathbf{Z} - \mathbf{h}(\mathbf{X}) \right\|^2$$

3. Use alternating optimisation to learn $\mathbf{h}$ and $\mathbf{Z}$ for each value of μ:
   - Over $\mathbf{Z}$: alternating optimisation over each bit.
   - Over $\mathbf{h}$: Learn a binary classifier for each bit independently.

Learning binary hashing: diversity-based approach

A recent method, Independent Laplacian Hashing (ILH) (Carreira-Perpiñán and Raziperchikolaei, NIPS 2016), proposed a diversity-based approach:

1. Learn output codes for each bit independently. This can be achieved by optimising a 1-bit objective function over the codes b times separately.
2. Learn a binary classifier for each bit independently.

To make the 1-bit hash functions different, the diversity techniques from the ensemble learning literature are used:

Different training sets for different bits, different subset of features for different 1-bit hash functions, etc.

This gives several advantages over the optimisation-based methods:

- Simpler and faster optimisation (over the 1-bit functions instead of the L-bits ones)
- Massive parallelism: The L-bit outputs of the hash functions can be trained independently.
- Better or comparable retrieval results to the previous approach.

Learning the hash function given the binary codes

In both approaches, a key step is to learn the hash function $\mathbf{h}$ that gives good binary codes:

- This corresponds to solving L binary classification problems independently: fit classifier l to the data ($\mathbf{X}$,$\mathbf{Z}_{\cdot,i}$) for $l = 1, \ldots, L$.
  ($\mathbf{X}_{D\times N} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ = images, $\mathbf{Z}_{\cdot,i} = (\mathbf{z}_{\cdot 1}, \ldots, \mathbf{z}_{\cdot N}) \in \{-1, 1\}^N$ = binary codes).
- Usually linear SVMs are used as the classifier, which gives a good hash function: The reason is that SVM gives a good classifier with a good generalization. Also, it solves a convex optimisation problem which is scalable to large training sets.

Putting the weights and biases of the binary classifiers together, we define the hash function as $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{W}\mathbf{x})$ where $\mathbf{W} \in \mathbb{R}^{L\times D}$.

- Generating binary codes for a query involves a matrix-vector multiplication which takes $\mathcal{O}(LD)$.

We can accelerate this by making $\mathbf{W}$ circulant (Yu et al. 2014)
and using the Fast Fourier Transform (FFT).

Hashing with a circulant weight matrix

A D-dimensional vector $\mathbf{w} = (w_0, w_1, \ldots, w_{D-1})$ is the basis for the $D \times D$ circulant matrix $\mathbf{W}$:

$$\mathbf{W} = \mathrm{circ}\,(\mathbf{w}) \equiv \left[\begin{array}{ccccc} w_0 & w_{D-1} & \cdots & w_2 & w_1 \\ w_1 & w_0 & w_{D-1} & \cdots & w_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ w_{D-1} & w_{D-2} & \cdots & w_1 & w_0 \end{array}\right] \Bigg\} \mathrm{circ}\,(\mathbf{w})_L$$

For $L < D$ bits, we only need the first L rows of $\mathrm{circ}\,(\mathbf{w})$: $\mathrm{circ}\,(\mathbf{w})_L$.

	Space Complexity	Time Complexity
Linear function	$\mathcal{O}(LD)$	$\mathcal{O}(LD)$
Circulant function	$\mathcal{O}(D)$	$\min(\mathcal{O}(LD), \mathcal{O}(D \log D))$

The reason is that the Discrete Fourier Transform $\mathcal{F}(\cdot)$ can be computed in $\mathcal{O}(D \log D)$ and the binary code is generated using DFT: $\mathbf{h}(\mathbf{x}) = \mathrm{sgn}\,(\mathbf{W}\mathbf{x}) = \mathrm{sgn}\left(\mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \circ \mathcal{F}(\mathbf{w}))\right)$.

If $L \gg \log D$, the circulant hash function can generate the codes faster than the linear one (non-circulant).

Yu et al. 2014 learn a circulant hash function with comparable results to the linear one. However, their learning algorithm is incorrect.

Circulant binary embedding (Yu et al. 2014)

Consider the dataset $\mathbf{X} \in \mathbb{R}^{D \times N}$ and the binary labels $\mathbf{Z} \in \{-1, 1\}^{L \times N}$.

CBE learns the circulant matrix $\mathbf{W} \in \mathbb{R}^{L \times D}$ to solve the classification problem as follows:

1. They pad the label matrix $\mathbf{Z}$ with $D - L$ zero rows to make it $D \times N$.
2. They solve the classification problem in the frequency domain.
   This involves a nonlinear optimisation over D independent problems in the complex plane.
3. They pick the first L rows of the resulting $\mathbf{W}$.

The padding step makes this algorithm incorrect, except for $L = D$.

For $L < D$, the resulting $\operatorname{circ}(\mathbf{w})_L$ is not the optimal solution.

As we make the L smaller, the error becomes larger.

Circulant support vector machines

We propose a correct way to learn the optimal circulant matrix.
Consider the dataset $\mathbf{X} \in \mathbb{R}^{D \times N}$ and the labels $\mathbf{Z} \in \{-1, 1\}^{L \times N}$.
We want to learn the circulant matrix $\mathbf{W} = \operatorname{circ}(\mathbf{w})_L \in \mathbb{R}^{L \times D}$ and the bias $\mathbf{b} \in \mathbb{R}^L$ that minimise the binary classification error

We consider the maximum margin formulation of the support vector machines (SVMs).

Consider $\mathbf{w}_l^T$ as the lth row of the matrix $\mathbf{W}$. The lth classification problem has the following form:

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

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.

The L problems are coupled because of $\mathbf{W} = \begin{pmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_L^T \end{pmatrix} = \operatorname{circ}(\mathbf{w})_L$.

Circulant support vector machines

Each of the L classification problems involves a circulantly rotated version of the vector $\mathbf{w}$. This is equivalent to L classification problems each with the same, unrotated $\mathbf{w}$, but with rotated input vector.
For example, consider the 2nd binary classification of a 3-D problem:

$$\begin{pmatrix} w_3 \\ w_1 \\ w_2 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}^T \begin{pmatrix} x_2 \\ x_3 \\ x_1 \end{pmatrix}$$

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. The SVM formulation of the lth classification problem becomes:

$$\min_{\mathbf{w}\in\mathbb{R}^D, b_l\in\mathbb{R}} \frac{1}{2}\left\|\mathbf{w}^T\mathbf{P}_l\right\|^2 + C\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad \begin{cases} z_{ln}(\mathbf{w}^T\mathbf{P}_l\mathbf{x}_n + b_l) \geq 1 - \xi_{ln} \\ \xi_{ln} \geq 0,\ n = 1,\ldots,N. \end{cases}$$

Since $\mathbf{P}_l^T\mathbf{P}_l = \mathbf{I}$, $\left\|\mathbf{w}^T\mathbf{P}_l\right\|^2 = \left\|\mathbf{w}\right\|^2$, so all L classification problems have the same margin term.

Circulant support vector machines

Let us define $\mathbf{t}_{ln} = \mathbf{P}_l \mathbf{x}_n \in \mathbb{R}^D$ and rewrite the objective function:

$$\min_{\mathbf{w}\in\mathbb{R}^D, b_l\in\mathbb{R}} \|\mathbf{w}\|^2 + \frac{2C}{L}\sum_{l=1}^{L}\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad \begin{cases} z_{ln}([\mathbf{w};\mathbf{b}]^T[\mathbf{t}_{ln};\mathbf{e}_l]) \geq 1-\xi_{ln}, \\ \xi_{ln} \geq 0,\ n = 1,\ldots,N, \\ l = 1,\cdots,L. \end{cases}$$

where $\mathbf{e}_l \in \mathbb{R}^L$ has 1 in the lth element and zeros everywhere else.

This is an SVM problem, with NL inputs $\mathbf{y}_{ln} = [\mathbf{t}_{ln};\mathbf{e}_l]$ and labels z_{ln}.

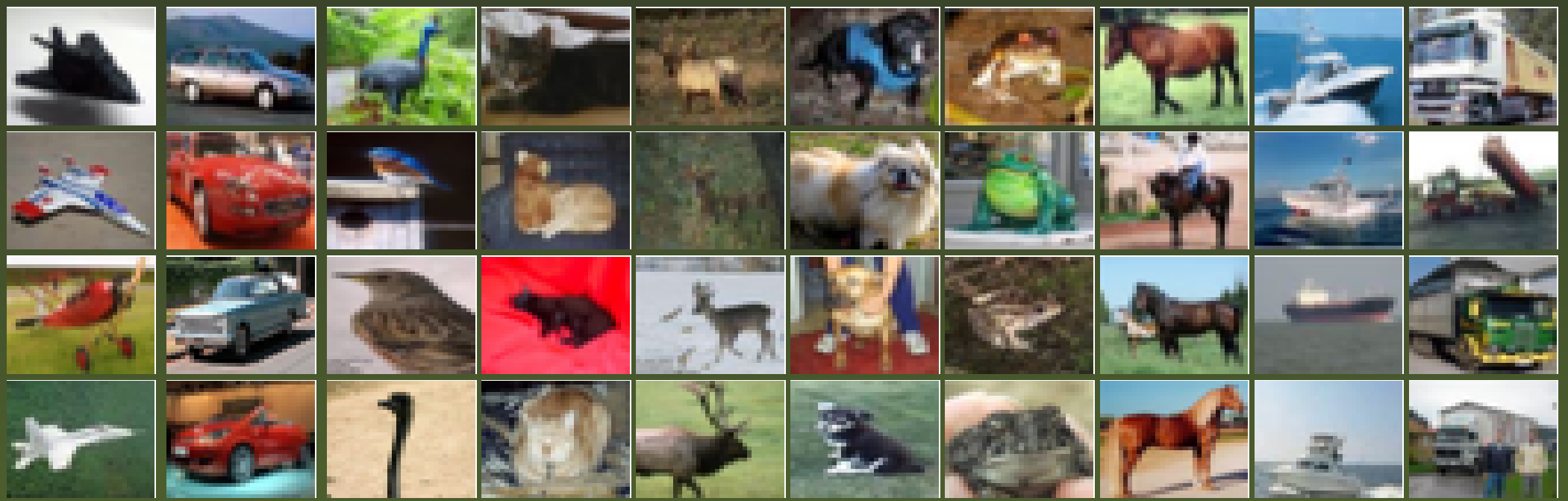
The only difference between our formulation and the SVM is that in here the margin is defined on a portion of weight vector. We use svmsgd from the VLFeat package and modify the computation of the gradient

- It always returns the optimal solution, even for the case of $L < D$.
- It is a convex quadratic program with a unique solution.
- There are libraries available that solve SVM problems for a large number of points in a few seconds.
- Our circulant SVM performs better than CBE in retrieval results.

Experimental setup: datasets and features

CIFAR-10 dataset. $60\,000$ 32×32 color images in 10 classes. We randomly select $58\,000/2\,000$ as the training/test set.

Features. Each image is represented by $D = 4\,096$-D VGG features: the output of the last fully connected layer of the VGG network.



Experimental setup: precision and recall

The performance of binary hash functions is usually reported using precision and recall.

Retrieved set for a query point can be defined in two ways:

- The K nearest neighbours in the Hamming space.
- The points in the Hamming radius of r.

Ground-truth for a query point contains the first K nearest neighbours of the point in the original(D-dimensional) space.

$$\text{recall} = \frac{|\{\text{retrieved points}\} \cap \{\text{groundtruth}\}|}{|\{\text{groundtruth}\}|}$$

$$\text{precision} = \frac{|\{\text{retrieved points}\} \cap \{\text{groundtruth}\}|}{|\{\text{retrieved points}\}|}$$

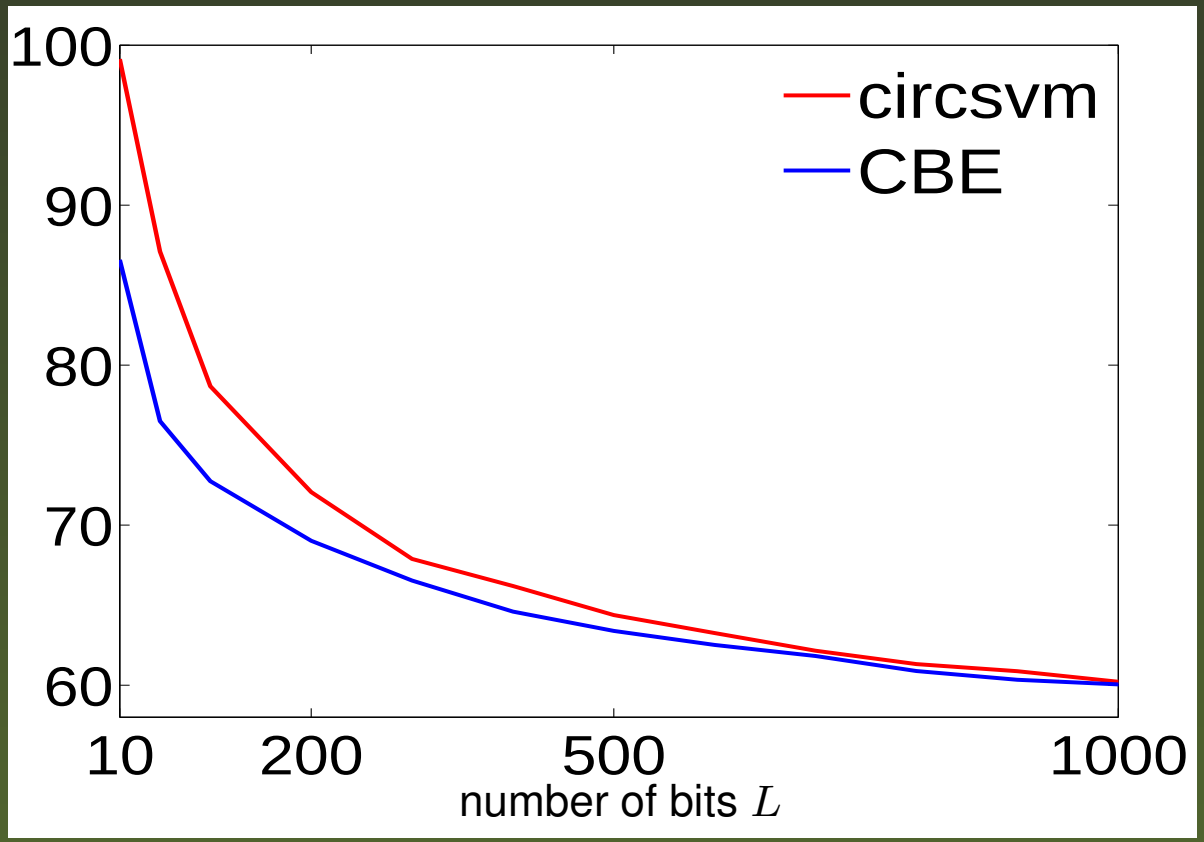
Circulant SVM improves the classification accuracy

We use the CIFAR images ($D = 4\,096$) as the input and the $L = 500$ binary codes generated by a hashing method (ITQ) as the labels.

We report the average accuracy of the L classification problems.

The circulant matrix is as a filter operating on the input image.

$f = 1$ filter, $L \geq 10$ bits



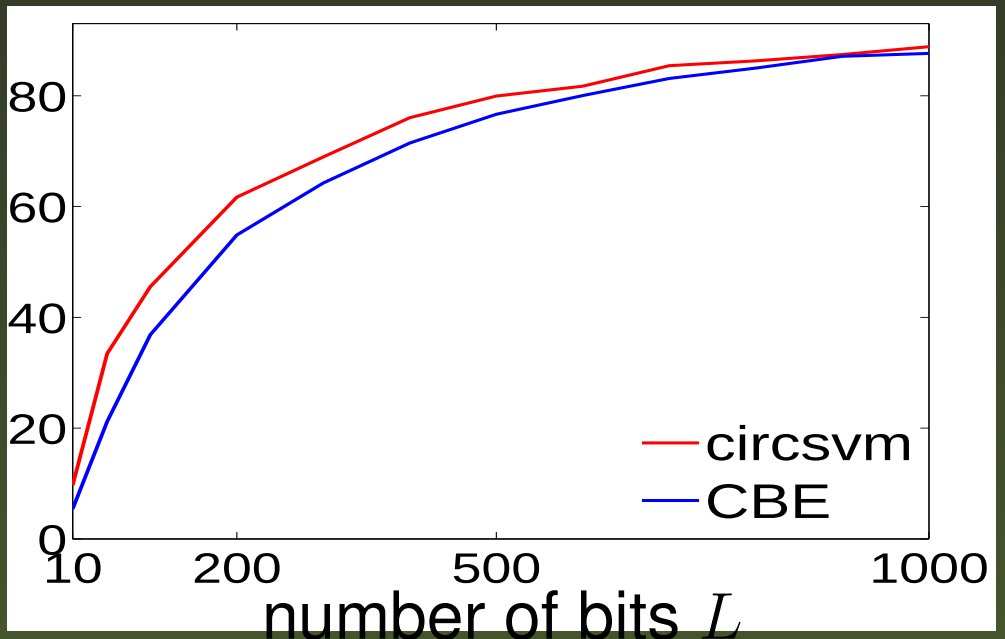
For smaller number of bits, CBE finds a suboptimal solution.

Our proposed method (Circulant SVM) always finds the optimal solution and gives a better classification accuracy.

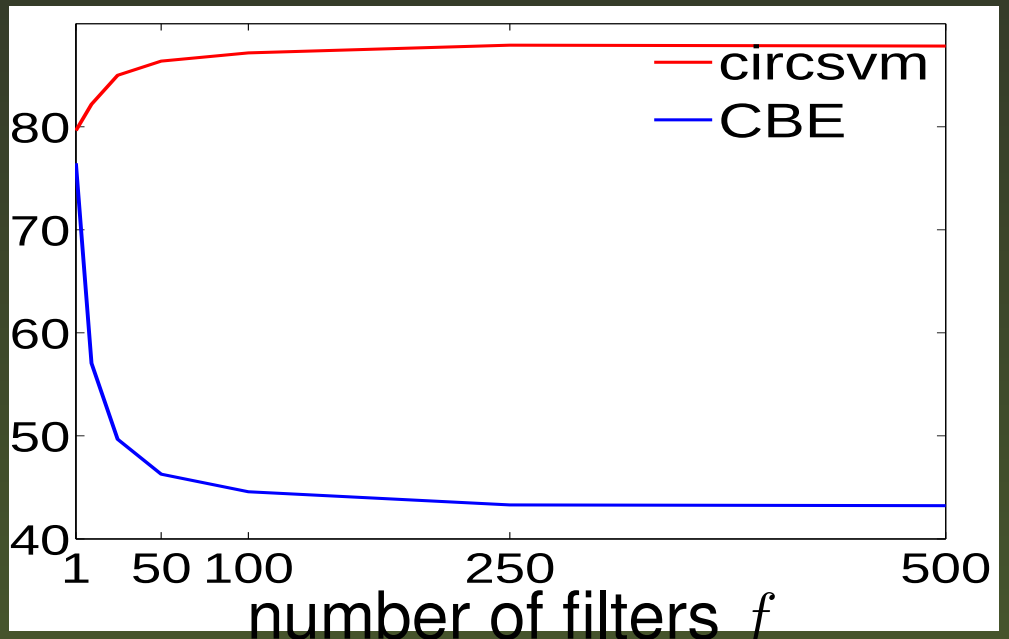
Circulant SVM improves the hashing results

We use the hash functions of the previous experiments in the hashing setting. We report recall for different number of bits and filters.

$f = 1$ filter, $L \geq 10$ bits



$L = 500$ bits, $f \geq 1$ filters



circsvm outperforms CBE. The improvement is more clear for smaller number of bits where CBE is unable to find the optimal solution.

For large f, CBE adds a massive number of zeros to the labels, loses the diversity among the functions, and performs much worse than circsvm.

Conclusion

- Using a circulant matrix as the weight matrix of a hash function makes the computation of the binary codes very fast.
- A previous work learns a suboptimal circulant matrix based on the optimization in the Fourier domain.
- We proposed a correct way 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 gives several advantages:
  - It always returns the optimal solution, even for $L < D$.
  - It is a convex quadratic program with a unique solution.
  - It can be implemented easily by reusing the libraries for SVMs.
  - This also results in learning better hash functions with better retrieval results.

Partly supported by NSF award IIS–1423515.