Kernel SVM Based on Binary Embedding and Optimized Random Fourier Features [1]

Zijian Lei, Liang Lan
Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China

Speaker: Apple Zhang

Shenzhen University

December 11, 2022

Table of Contents

Table of Contents

Kernel SVM

Consider the following nonlinear SVM

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + c\sum_{i=1}^{n}[1 - y_i\mathbf{w}^\top\phi(\mathbf{x}_i)]_+, \tag{1}$$

where $\phi(\cdot)$ is a nonlinear mapping which is implicitly defined by the kernel function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle \,. \tag{2}$$

However, the storage of kernel matrix is expensive. It need $O(n^2)$ space.

Random Fourier Feature

We aims to find $\mathbf{z}(\mathbf{x})$ such that

$$k(\mathbf{x}_i, \mathbf{x}_j) \approx \mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j). \tag{3}$$

Random Fourier Feature (RFF) [2] is an efficient way to approximate the kernel features. Consider the shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ and its Fourier transform:

$$k(\mathbf{x} - \mathbf{y}) = \int_{\mathbb{R}^d} p(\mathbf{w}) \exp\left(i\mathbf{w}^\top(\mathbf{x} - \mathbf{y})\right) \mathrm{d}\mathbf{w} = \mathbb{E}_{\mathbf{w}\sim p(\mathbf{w})}[\zeta_{\mathbf{w}}(\mathbf{x})\bar{\zeta}_{\mathbf{w}}(\mathbf{y})]. \tag{4}$$

Then we can construct $\mathbf{z}(\mathbf{x})$ as

$$\mathbf{z}(\mathbf{x}) = \sqrt{\frac{2}{D}} \cos\left(\mathbf{W}^\top \mathbf{x} + \mathbf{b}\right), \quad \mathbf{W} \in \mathbb{R}^{d\times D},\ \mathbf{b} \in \mathbb{R}^D. \tag{5}$$

where $\mathbf{w} \sim p(\mathbf{w})$ and $b_i \sim \mathrm{Uniform}(0, 2\pi)$.

Binary Codes for the Shift-Invariant Kernels (BCSIK)

To further reduce the storage burden, BCSIK obtain the binarized RFF that can be store by bits.

$$\mathbf{z}(\mathbf{x}) = \mathrm{sign}(\cos(\mathbf{W}^\top \mathbf{x} + \mathbf{b})) \in \{-1, 1\}^D. \tag{6}$$

Storage space: $(32 \times D)$-bits $\rightarrow$ D-bits

Lemma

Define

$$h_1(u) = \frac{4}{\pi^2}(1-u), \quad h_2(u) = \min\left\{\frac{1}{2}\sqrt{1-u^2}, \frac{4}{\pi^2}\left(1-\frac{2}{3}u\right)\right\}. \tag{7}$$

Fixing $\delta, \epsilon \in (0,1)$, *for any dataset* $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, *with probability at least* $1-\epsilon$ *and* $p \geq \log(n^2/\epsilon)/(2\delta^2)$, *we have*

$$h_1(k(\mathbf{x}_i, \mathbf{x}_j)) - \delta \leq d_H(\mathbf{z}_i, \mathbf{z}_j)/D \leq h_2(k(\mathbf{x}_i, \mathbf{x}_j)) + \delta. \tag{8}$$

Table of Contents

Reduced Memory BCSIK

Use Fastfood method [3] to obtain the RFF.

$$\mathbf{V}_i^\top = \frac{1}{\sigma\sqrt{d}}\mathbf{SHG\Pi HB}. \tag{9}$$

We later explain the notations. Let $\mathbf{W} = [\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_{p/d}]$. Therefore, we can compute

$$\mathbf{z}^{(i)} = \mathbf{V}_i^\top \mathbf{x} = \frac{1}{\sigma\sqrt{d}}\mathbf{SHG\Pi HBx}. \tag{10}$$

This computation costs $O(d \log d)$ time and $O(d)$ space. Then we have

$$\mathbf{W}^\top \mathbf{x} = [\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(p/d)}], \tag{11}$$

and compute the binary embedding $\mathbf{z}(\mathbf{x}) = \mathrm{sign}(\cos(\mathbf{W}^\top \mathbf{x} + \mathbf{b}))$.

Explanation of Matrices

$$\mathbf{V}_i^\top = \frac{1}{\sigma\sqrt{d}}\mathbf{S}\mathbf{H}\mathbf{G}\mathbf{\Pi}\mathbf{H}\mathbf{B}. \tag{12}$$

- $\mathbf{S}$: diagonal matrix and $s_{ii} \sim \mathrm{Uniform}(0, 1)$.
- $\mathbf{G}$: diagonal matrix and $g_{ii} \sim \mathcal{N}(0, 1)$.
- $\mathbf{B}$: diagonal matrix and $b_{ii} \sim \mathrm{Rad}$, which is Rademacher random variable:

$$p(\sigma = -1) = p(\sigma = 1) = \frac{1}{2}. \tag{13}$$

- $\mathbf{\Pi}$: random permutation matrix.
- $\mathbf{H}$: Walsh-Hadamard matrix, defined as

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{14}$$

Note that $\mathbf{H}\mathbf{x}$ can be computed via Fast Walsh-Hadamard Transform (FWHF), costing $O(d \log d)$ time.

Table of Contents

My Improvement

We can improve the original kernel into a new kernel k_Q [4].

$$k_Q(\mathbf{x}, \mathbf{x}') = \int \phi(\mathbf{x}; \mathbf{w})\phi(\mathbf{x}'; \mathbf{w})\mathrm{d}Q(\mathbf{w}). \tag{15}$$

Then we do the kernel alignment via

$$\max_{Q \in \mathcal{P}} \sum_{i,j} k_Q(\mathbf{x}_i, \mathbf{x}_j) y_i y_j. \tag{16}$$

We have the following emprical version of kernel alignment.

$$\max_{\mathbf{q} \in \bar{\mathcal{P}}} \sum_{i,j} y_i y_j \sum_{l=1}^{N_w} q_l \phi(\mathbf{x}_i; \mathbf{w}_l)\phi(\mathbf{x}_j; \mathbf{w}_l). \tag{17}$$

where $y_i \in \{-1, 1\}$ is label and $\bar{\mathcal{P}} = \{\mathbf{q} : D_f(\mathbf{q}||\mathbf{1}/N_w) \leq \rho\}$.

My Improvement

f-Divergence is defined as

$$D_f(Q||P) = \int f\left(\frac{p(\mathbf{w})}{q(\mathbf{w})}\right) q(\mathbf{w})\mathrm{d}\mathbf{w}. \tag{18}$$

Using $f(t) = t^2 - 1$. We can obtain the optimal $\mathbf{q}$ via

$$\max_{\mathbf{q}\in\Delta} \mathbf{q}^\top \mathbf{v} - \frac{\lambda}{N_w}\sum_{l=1}^{N_w}(N_w q_l)^2, \tag{19}$$

where $\Delta = \{\mathbf{q} \in \mathbb{R}_+^{N_w} : \mathbf{q}^\top \mathbf{1} = 1\}$, and $\lambda \geq 0$ is the Lagrange multiplier which is solved by dual problem.

Improvement for multi-class case

Define

$$\bar{y}_{ij} = \begin{cases} 1, & y_i = j, \\ -1, & \text{otherwise.} \end{cases} \tag{20}$$

Then we can redefine the objective as

$$\sum_{i,j} \bar{\mathbf{y}}_i^\top \bar{\mathbf{y}}_j \sum_{l=1}^{N_w} q_l \phi(\mathbf{x}_i; w_l)\phi(\mathbf{x}_j; w_l) = \sum_{l=1}^{N_w} q_l \left\| \sum_{i=1}^{n} \bar{\mathbf{y}}_i \phi(\mathbf{x}_i; w_l) \right\|^2 \tag{21}$$

$$= \mathbf{q}^\top \mathbf{v}. \tag{22}$$

where $v_l = \|\sum_{i=1}^n \bar{\mathbf{y}}_i \phi(\mathbf{x}_i; w_l)\|^2$.

Advantages

- Data-dependent learning. Use label information to improve the nonlinear features.
- Reduce storage burden. The solution of (17) is sparse, which means the random features with zero probability are eleminated.
- Can be easily implemented based on RM-BCSIK.

Table of Contents

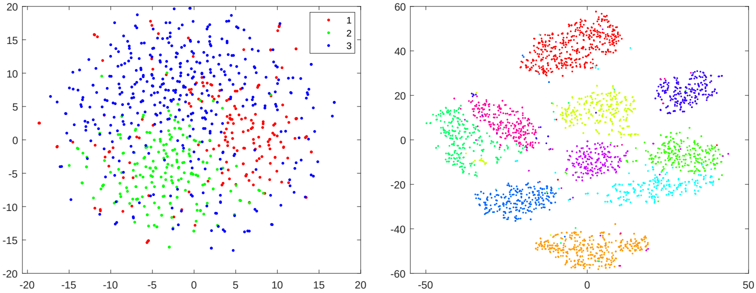
Experimental results

Table: The classification accuracy of different nonlinear SVM on different datasets.

Datasets	RFF+SVM	this paper	ours	exact
Sonar	*82.89*±4.99	80.84±5.45	**83.49**±2.90	82.05±5.37
DNA	*93.48*±1.94	91.84±3.92	92.31±0.63	**94.25**±0.55
BinaryAlpha	68.79±2.93	66.69±3.68	*70.07*±1.14	**70.89**±1.66
USPS	**96.54**±0.29	95.36±0.39	95.73±0.25	*96.44*±0.24

- The RM-BCSIK method reach the promising performance.
- Our method outperforms RM-BCSIK and its performance has no significant difference with the exact kernel SVM.

Experimental results



(a) DNA (b) USPS

Figure: The visualization of the learned binary codes.

Thank you!

[1] Z. Lei and L. Lan, “Memory and computation-efficient kernel svm via binary embedding and ternary model coefficients,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 8316–8323, 2021.

[2] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” *Advances in Neural Information Processing Systems*, vol. 20, 2007.

[3] Q. Le, T. Sarlós, A. Smola, *et al.*, “Fastfood-approximating kernel expansions in loglinear time,” in *Proceedings of the International Conference on Machine Learning*, vol. 85, p. 8, 2013.

[4] A. Sinha and J. C. Duchi, “Learning kernels with random features,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.