Supplementary Material for "Prototype Discriminative Learning for Image Set Classification"

Wen Wang, Ruiping Wang, Shiguang Shan, Xilin Chen

[1]Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China

[2]University of Chinese Academy of Sciences, Beijing, 100049, China

wen.wang@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

I. Gradient derivation

In this supplementary material, a detailed derivation is given for the gradient computing formulas of Equ. (8) and (9) in Sec. II-C. As defined in Equ. (5) in Sec. II-B, the objective function can be rewritten as follows.

$$J(W, P_1, .., P_C) = \sum_{c=1}^{C} \sum_{x \in X_c} \mathcal{S}_\beta(Q_x), \tag{S1}$$

where

$$\mathcal{S}_\beta(z) = \frac{1}{1 + e^{\beta(1-z)}}, \tag{S2}$$

and

$$Q_x = \frac{d(y, nn_w^c(y))}{d(y, nn_b^c(y))}. \tag{S3}$$

Note that $d(\cdot, \cdot)$ denotes the Euclidean distance, i.e., for any $y, z \in \mathbb{R}^r$,

$$d(y, z) = \sqrt{\sum_{j=1}^{r} (y_j - z_j)^2}. \tag{S4}$$

As claimed in Sec. II-C, we assume that the same prototype neighbor is searched when the variation in the prototype sets and transformation matrix is sufficiently small. Under such assumption, we can derive the gradient of loss function J with respect to W approximately according to the Chain Rule:

$$\begin{aligned}\frac{\partial J}{\partial W_k} \approx & \sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_b^c(y))}\cdot\frac{\partial d(y,nn_w^c(y))}{\partial W_k}d(y,nn_b^c(y)) \\ & -\sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_b^c(y))}\cdot d(y,nn_w^c(y))\frac{\partial d(y,nn_b^c(y))}{\partial W_k}.\end{aligned} \tag{S5}$$

We can easily get

$$\begin{aligned}\frac{\partial d(y,nn_w^c(y))}{\partial W_k} &= \frac{2(x-a)(y_k-nn_w^c(y)_k)}{2\sqrt{\sum_{j=1}^{r}(y_j-nn_w^c(y)_j)^2}} = \frac{(x-a)(y_k-nn_w^c(y)_k)}{d(y,nn_w^c(y))}, \\ \frac{\partial d(y,nn_b^c(y))}{\partial W_k} &= \frac{2(x-b)(y_k-nn_b^c(y)_k)}{2\sqrt{\sum_{j=1}^{r}(y_j-nn_b^c(y)_j)^2}} = \frac{(x-b)(y_k-nn_b^c(y)_k)}{d(y,nn_b^c(y))}.\end{aligned} \tag{S6}$$

By substituting Equ. (S6) into Equ. (S5), Equ. (8) can be finally derived as follows.

$$\begin{aligned}\frac{\partial J}{\partial W_k} \approx & \sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_w^c(y))}\cdot\frac{d(y,nn_w^c(y))}{d(y,nn_b^c(y))}\cdot(x-a)(y_k-nn_w^c(y)_k) \\ & -\sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_b^c(y))}\cdot\frac{d(y,nn_w^c(y))}{d(y,nn_b^c(y))}\cdot(x-b)(y_k-nn_b^c(y)_k) \\ = & \sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)Q_x}{d^2(y,nn_w^c(y))}\cdot(x-a)(y_k-nn_w^c(y)_k) \\ & -\sum_{c=1}^{C}\sum_{x\in X_c}\frac{\mathcal{S}'_\beta(Q_x)Q_x}{d^2(y,nn_b^c(y))}\cdot(x-b)(y_k-nn_b^c(y)_k),\end{aligned} \tag{S7}$$

Similarly, we derive the gradient of J with respect to each vector $v_{ci}\in\mathbb{R}^{l_c}$, $i=1,...,m_c$, $c=1,...,C$.

$$\begin{aligned}\frac{\partial J}{\partial v_{ci}} \approx & \sum_{c=1}^{C}\sum_{\substack{x\in X_c\\ p_{ci}=a}}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_b^c(y))}\cdot\frac{\partial d(y,nn_w^c(y))}{\partial v_{ci}}d(y,nn_b^c(y)) \\ & -\sum_{c=1}^{C}\sum_{\substack{x\in X_c\\ p_{ci}=b}}\frac{\mathcal{S}'_\beta(Q_x)}{d^2(y,nn_b^c(y))}\cdot d(y,nn_w^c(y))\frac{\partial d(y,nn_b^c(y))}{\partial v_{ci}}.\end{aligned} \tag{S8}$$

Since we can easily obtain

$$\begin{aligned}\frac{\partial d(y,nn_w^c(y))}{\partial v_{ci}} &= \frac{U_c^TWW^T(a-x)}{d(y,nn_w^c(y))}, \\ \frac{\partial d(y,nn_b^c(y))}{\partial v_{ci}} &= \frac{U_c^TWW^T(b-x)}{d(y,nn_b^c(y))},\end{aligned} \tag{S9}$$

we can finally deduce Equ. (9)

Algorithm 1 Optimization algorithm for PDL-OP

Input:

Data matrices of C image sets for training: $\{X_1, X_2, ..., X_C\}$ and their labels;

the slope for sigmoid function: β;

the initial prototype sets: $P = \{P_1, ..., P_C\}$;

the initial transformation matrix: W.

Output:

The optimal P and W

1: **while** not converged **do**

2: Compute the loss function according to Equ. (5);

3: Generate A according to Equ. (S10);

4: Compute the step size τ for searching W by the curvilinear searching algorithm [1]. Call line search along the path $W(\tau)$ defined by Equ. (S11);

5: Compute the update of V by the L-BFGS algorithm;

6: Update V and W;

7: **end while**

8: Compute P by affine coefficients V according to Equ. (3);

9: **return** P^*, W^*;

II. Optimization algorithm

To solve the optimization problem $\min_{W,V} J(W, V)$ without projection constraint, we can update W and V in an iterative procedure based on the derived gradients above by using limited-memory BFGS (L-BFGS) [2] to control the step size.

As for the optimization problem $\min_{W,V} J(W, V)$ with orthonormal projection constraint $W^T W = I_r$, it is non-trivial to jointly optimize W and V as the feasible set of W is on a Stiefel manifold. Considering its favorable property of low computational cost, we choose the curvilinear searching method in [1] to search the step size and the path of W.

Formally, according to [1], we define a skew-symmetric matrix A as

$$A = GW^T - WG^T. \tag{S10}$$

where $G = \frac{\partial J}{\partial W}$. To drive the new point to satisfy $W'^T W' = I_r$, it is searched along a curve $W(\tau)$ given

by

$$W(\tau) = W - \tau A(\frac{W + W(\tau)}{2}), \tag{S11}$$

where τ is a step size and $W(\tau)$ satisfies $W(\tau)^T W(\tau) = W^T W$.

Furthermore, to guarantee the optimization of W and V to be performed jointly and consistently, at each step, we update W along the curve $W(\tau)$ by using [1] and search the new iteration of V along straight lines by using L-BFGS. The optimization algorithm is summarized in Alg. 1.

REFERENCES

[1] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, 2013.

[2] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *International Conference on Machine learning (ICML)*, 2011, pp. 265–272.