Self-Supervised Deep Low-Rank Assignment Model for Prototype Selection

Xingxing Zhang, Zhenfeng Zhu, Yao Zhao, Deqiang Kong
Institute of Information Science, Beijing Jiaotong University, Beijing, China
Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing, China
{zhangxing,zhfzhu,yzhao,dqkong}@bjtu.edu.cn

Abstract

Prototype selection is a promising technique for removing redundancy and irrelevance from large-scale data. Here, we consider it as a task assignment problem, which refers to assigning each element of a source set to one representative, i.e., prototype. However, due to the outliers and uncertain distribution on source, the selected prototypes are generally less representative and interesting. To alleviate this issue, we develop in this paper a Self-supervised Deep Low-rank Assignment model (SDLA). By dynamically integrating a low-rank assignment model with deep representation learning, our model effectively ensures the goodness-of-exemplar and goodness-of-discrimination of selected prototypes. Specifically, on the basis of a denoising autoencoder, dissimilarity metrics on source are continuously self-refined in embedding space with weak supervision from selected prototypes, thus preserving categorical similarity. Conversely, working on this metric space, similar samples tend to select the same prototypes by designing a low-rank assignment model. Experimental results on applications like text clustering and image classification (using prototypes) demonstrate our method is considerably superior to the state-of-the-art methods in prototype selection.

1 Introduction

Prototype selection is the task of finding exemplar samples, called prototypes, from a large collection of data points. This is at the center of many problems in data analysis and processing field because it holds several advantages over data storage, compression, synthesis, cleansing and visualization. First, the memory cost for storing information on the data can be significantly reduced using prototypes. Second, prototypes help in clustering of data, and, as the most prototypical samples, can be used for efficient synthesis of new data points. More importantly, the computational efficiency for data modeling, such as classifier training [Garcia et al., 2012; Zhang et al., 2018] and active learning [Lin et al., 2018], can be improved by working on prototypes. In addition, selecting prototypes helps to remove redundant or irrelevant points, such as outliers. Finally, prototype selection has been applied for anomaly detection [Cong et al., 2011], web media summarization and recommendation [Meng et al., 2016; Cong et al., 2017] (see Figure 1), segmentation of dynamic data [Elhamifar et al., 2016], and more.

To characterize the informativeness of prototypes in terms of ability to represent the entire distribution, many selection strategies have been proposed, generally including two categories. The first one involves greedy methods that select next prototype with respect to the previously selected prototypes, and usually maximizes submodular functions, such as graph-cuts and facility location [Wang and Zhang, 2013; Elhamifar and Kaluza, 2017; Krause et al., 2008]. Obviously, they usually result in high computational complexity. The second category is comprised of model-driven methods that aim to maximize global objective based on subspace leaning or pairwise relationship. Their resulting solutions are closely related to a variety of optimization algorithms.

Concretely, subspace leaning based models mainly focus on the data that lie in one or several low-dimensional subspaces. The Rank Revealing QR algorithm [Chan, 1987] is a representative one. By representing data in a low dimensional space with possibly minimal representation error, various effective models have been formulated to learn a selection matrix, such as the sparse dictionary selection (SDS) method [Cong et al., 2011] and sparse modeling representative selection (SMRS) model [Elhamifar et al., 2012].
To further exploit data structure, some improved methods have been proposed [Dornaika and Aldine, 2015; Wang et al., 2016]. For instance, structured sparse dictionary selection (SSDS) [Wang et al., 2017] method tried to select prototypes with both representativeness and diversity via three structured regularizers. However, by multi-linear coding criteria, these methods are effective only when the samples from different groups are sufficiently dissimilar.

In contrast, pairwise relationship based models aim at the data that could be naturally grouped under a certain dissimilarity metric. One naive approach is Kmedoids [Kaufman and Rousseeuw, 1987], which finds $k$ medoid centers as prototypes by pursuing the minimum total distance from all samples. Some variants [Nellore and Ward, 2015] were further proposed based on Kmedoids. Unlike Kmeans [Duda et al., 2012] or Kmedoids, Affinity Propagation (AP) algorithm [Frey and Dueck, 2007] does not require any initialization for prototypes, but has suboptimal property. To tackle this issue, dissimilarity-based sparse subset selection (DS3) method [Elhamifar et al., 2016] was recently proposed to select prototypes via a trace minimization model. In general, this kind of methods performs well only under appropriate similarity metric.

Furthermore, some real datasets do not live in a vector space, e.g., social network data or proteomics data. Therefore, model-driven methods based on pairwise relationship have more advantages on prototype selection. However, most existing algorithms suffer from imposing restrictions on the type of pairwise relationship. More importantly, unreliable metric caused by outliers or uncertain data distribution significantly reduces the quality of prototypes. Inspired by representation learning [Song et al., 2017], we consider introducing a unified framework for dissimilarity learning and prototype selection. As a result, the metric is efficiently self-refined with weak supervision from prototypes, and conversely, prototypes are more informative by using refined metrics. These construct the basic motivation of our framework.

In summary, the main contributions of this work are highlighted as follows.

- By considering the prototype selection as a task assignment problem, we develop a Self-supervised Deep Low-rank Assignment model (SDLA), which aims to jointly learn ideal dissimilarity metrics in embedding space and select informative prototypes in metric space.

- Unlike DS3 [Elhamifar et al., 2016], inspired by the good performance of deep representation learning, we propose to learn dissimilarity metrics based on a variant of a denoising autoencoder, in which categorical similarity is preserved with weak supervision from prototypes.

- Working on the learned metric space, similar samples tend to select the same prototypes by designing a low-rank assignment model, thus guaranteeing a diversified selection.

- The quality of prototypes selected by the proposed framework is reasonably evaluated with examples in text clustering and image classification (using prototypes), showing very promising results.

2 The Proposed Framework

2.1 Problem Statement

Let $X = \{x_1, \ldots, x_m\} \in \mathbb{R}^{d \times m}$ be the source data matrix of $m$ samples in $\mathbb{R}^d$. As shown in Figure 2, we consider the problem of selecting an interesting subset of $m$ samples, called prototype set, that can efficiently describe and summarize all samples in source data. Inspired by DS3 [Elhamifar et al., 2016], we primarily pursue the minimum assignment cost based on pairwise relationships on source.

2.2 Problem Formulation

It is noteworthy that pairwise dissimilarity matrix $D = \{d_{ij}\}_{i=1,\ldots,m}^{j=1,\ldots,m}$ between samples is directly given as the input to DS3 [Elhamifar et al., 2016]. Actually, the given $d_{ij}$, such as Euclidean distance or $\chi^2$ distance, may be not discriminative due to uncertain distribution and outliers in $X$, thus leading to a less representative and diversified selection. In light of the good performance of representation learning [Bengio et al., 2013], we consider circularly refining the dissimilarity metric and the selection indicator. Thereby, available semantic information from selected prototypes can be employed to learn the parameters of metric architecture, while conversely, more discriminative metric can improve the quality of prototypes. Towards this end, we develop a Self-supervised Deep Low-rank Assignment model (SDLA) for prototype selection as follows:

$$\min_{\Theta, \{z_{ij}\}} \sum_{j=1}^{m} \sum_{i=1}^{m} \Phi_{dSim}(\Phi_{rep}(\Theta; x_i), \Phi_{rep}(\Theta; x_j)) z_{ij}$$

$$+ \omega_1 \Omega(\Theta) + \omega_2 \Psi(Z)$$

subject to

$$\sum_{i=1}^{m} z_{ij} = 1, \forall j; \quad z_{ij} \geq 0, \forall i, j,$$

where $\Phi_{rep}(\Theta; \cdot)$ learns a deep representation of a sample, and $\Theta$ is the set of related parameters. $\Phi_{dSim}(\cdot; \cdot)$ aims to capture the dissimilarity of two samples in learned embedding space, thus $d_{ij} = \Phi_{dSim}(\Phi_{rep}(\Theta; x_i), \Phi_{rep}(\Theta; x_j))$. $Z = \{z_{ij}\}_{i=1,\ldots,m}^{j=1,\ldots,m}$ is a selection matrix, $z_{ij} \in [0,1]$ is the probability of selecting $x_i$ as the prototype of $x_j$, thus enforcing $\sum_{i=1}^{m} z_{ij} = 1$ for $x_j$. The first term in the objective function corresponds to the total assignment cost, the second term corresponds to the constraints in the deep architecture,
and the last term corresponds to the number of prototypes. \( \omega_1 \) and \( \omega_2 \) are nonnegative parameters to balance these terms. Figure 3 provides an illustration of the proposed model in (1).

Concretely, for any \( x_i \) and \( x_j \), if \( d_{ij} = 0 \), then \( z_{ij} = 1 \) due to low assignment cost, which implies that \( x_i \) and \( x_j \) are from the same class. While \( z_{ij} = 0 \) if \( d_{ij} = \infty \), which implies that \( x_i \) and \( x_j \) are from different classes. Since \( Z \) provides additional semantic information for all samples, \( D \) will be learned more discriminatively by preserving categori-
al similarity with weak supervision from \( Z \).

Considering the separability of both objective and constraints in (1), we further divide (1) into two subproblems \( P_1 \) and \( P_2 \), which are cyclic self-refinement:

- \( P_1 \) corresponding to the pairwise dissimilarity learning module in Figure 3, is written as

\[
\min_{\Theta} \sum_{i,j=1}^{m} \Phi_{dSim}(\Phi_{rep}(\Theta; x_i), \Phi_{rep}(\Theta; x_j)) z_{ij} + \omega_1 \Omega(\Theta)
\]

(2)

- \( P_2 \) corresponding to the assignment model learning module in Figure 3, is written as

\[
\min_{\{z_{ij}\}} \sum_{i,j=1}^{m} \Phi_{dSim}(\Phi_{rep}(\Theta; x_i), \Phi_{rep}(\Theta; x_j)) z_{ij} + \omega_2 \Psi(Z)
\]

s.t. \( \sum_{i=1}^{m} z_{ij} = 1, \forall j; \ z_{ij} \geq 0, \forall i, j, \)

(3)

The details about learning \( \Theta \) and \( Z \) are introduced in Section 3.1 and Section 3.2, respectively. As a result, the indices of nonzero rows of the solution \( Z^* \) correspond to the indices of those samples that are chosen as the data prototypes. In addition, \( Z^* \) shows the membership of samples in \( X \) to prototypes. That is, \( z_{ij}^* = [z_{ij}, \ldots, z_{mj}]^T \in \mathbb{R}^m \) corresponds to the probability vector of \( x_j \) being represented by each sample in \( X \). Therefore, we can obtain a partitioning of \( X \) under the rule that, if \( X = \{x_1, \ldots, x_m\} \) denotes the set of selected prototypes, we can assign \( x_i \) to the prototype \( x_{\delta_j} \) by

\[
\delta_j = \arg \min_{i \in \{1, \ldots, m\}} z_{ij}
\]

Consequently, \( X \) is classified into \( \kappa \) groups corresponding to \( \kappa \) prototypes via the selection matrix \( Z^* \).

3 Optimization

3.1 Pairwise Dissimilarity Learning

As shown in (2), optimizing \( \Theta \) aims to refine pairwise dissimilarities \( \{d_{ij}\} \), and it seamlessly connects the visual content and dissimilarity metric with weak supervision from \( Z \). However, due to the outliers and uncertain distribution on \( X \), we consider generating distributed representation vectors and pairwise dissimilarities on the basis of a denoising autoencoder [Vincent et al., 2008]. The general denoising autoencoder is formulated as:

\[
\begin{align*}
\tilde{x}_i &\sim q(\tilde{x}_i|x_i); \\
s_i &= f(W \tilde{x}_i + b); \\
y_i &= f(W^* s_i + b'); \\
L_R(y_i, x_i) &= \|x_i - y_i\|_2; \\
\Theta &= \arg \min_{W, W^*, b, b'} \sum_{i=1}^{m} L_R(y_i, x_i).
\end{align*}
\]

where \( x_i \in X \) is the original input vector, \( i = 1, \ldots, m \), and \( q(\cdot) \) is the corrupting distribution. The stochastically corrupted vector, \( \tilde{x}_i \), is obtained from \( q(\cdot|x_i) \). Generally, \( \tilde{x}_i = (1 - \beta) x_i \), and \( \beta \) is the corruption rate in the training phase. The hidden representation, \( s_i \), is mapped from \( \tilde{x}_i \) through the network, which consists of an activation function \( f(\cdot) \), parameter matrix \( W \), and parameter vector \( b \). In the
Finally, we use the elementwise sigmoid function as \( f(\cdot) \), and masking noise as \( q(\cdot) \). The following model is trained by mini-batch Adaptive Moment Estimation:

\[
\Theta = \arg\min_{W_i, W_i', b_i, b_i'} L_A(X) + \omega_1(\Theta) \tag{10}
\]

where \( \Omega(\Theta) = \sum_{l=1}^{L/2} (||W_l||^2_F + ||b_l||_2^2 + ||W_l'||^2_F + ||b_l'||_2^2) \), and \( l = 1, \ldots, L/2 \).

### 3.2 Assignment Model Learning

As shown in (3), optimizing \( Z \) aims to select a small number of prototypes with both diversity and representativeness. Then we consider enforcing the lowest rank and sparsity properties on \( Z \). That is, \( \Psi(Z) = \lambda_1 \text{rank}(Z) + \lambda_2 \text{card}(Z) \), where \( \lambda_1, \lambda_2 > 0 \), \( \text{rank}(Z) \) denotes the rank of \( Z \), and \( \text{card}(Z) \) denotes the number of nonzero elements of \( Z \). As observed in [Zhuang et al., 2012], the low-rankness criterion is better at capturing the global structure of dissimilarity \( D \), while the sparsity criterion can capture the local structure of each data vector. Specifically, low-rankness can encourage similar samples to have similar codes (i.e., \( i^{th} \) and \( j^{th} \) columns of \( Z \)), so as to select the same prototypes. Thus, (3) is rewritten as

\[
\min_Z \text{tr}(D^T Z) + \lambda_1 ||Z||_s + \lambda_2 ||Z||_0 \tag{11}
\]

subject to \( 1^T Z = 1^T; \ Z \geq 0 \),

where \( \sum_{i,j=1}^m d_{ij} z_{ij} = \text{tr}(D^T Z) \), \( \text{tr}(\cdot) \) denotes the trace operator, and \( d_{ij} \) has been obtained from (2). \( ||\cdot||_s \) and \( ||\cdot||_0 \) are the nuclear norm and \( \ell_0 \)-norm of a matrix, respectively. \( \lambda_1 \) and \( \lambda_2 \) are two nonnegative regularization parameters to balance these terms. \( 1 \in \mathbb{R}^m \) is a column vector with all 1’s.

The problem in (11) could be solved by Inexact Augmented Lagrangian Method (IALM). It is an iterative algorithm, and thus needs to first introduce two auxiliary variables \( Z_1 \) and \( Z_2 \) to make the objective function separable. The following problem is obtained:

\[
\min_{Z, Z_1, Z_2} \text{tr}(D^T Z) + \lambda_1 ||Z||_s + \lambda_2 ||Z||_0 \tag{12}
\]

subject to \( Z_1 = Z; \ Z_2 = Z; \ 1^T Z = 1^T; \ Z_2 \geq 0 \),

where \( ||\cdot||_1 \) is the \( \ell_1 \)-norm of a matrix to relax the \( ||\cdot||_0 \). Then, the augmented Lagrangian function of (12) is as follows:

\[
\mathcal{L}(Z, Z_1, Z_2) = \text{tr}(D^T Z) + \lambda_1 ||Z||_1 + \lambda_2 ||Z||_0 + (\Delta_1, Z - Z_1) + (\Delta_2, Z - Z_2) + \langle \delta_3, 1^T Z - 1^T \rangle + \frac{\eta}{2} \left( ||Z - Z_1||_F^2 + ||Z - Z_2||_F^2 + ||1^T Z - 1^T||_2^2 \right) \tag{13}
\]

where \( \Delta_1, \Delta_2 \in \mathbb{R}^{m \times m} \) and \( \delta_3 \in \mathbb{R}^m \) are the Lagrange multipliers. \( \eta \) is a penalty parameter.

According to the IALM, the objective function converges with a sequence of closed form updating steps. The variable \( Z, Z_1, \) or \( Z_2 \) is updated with other variables fixed. The detailed updating rules are presented as follows.

\[
Z = \text{argmin}_Z \mathcal{L}(Z) = (2\eta I + \eta 1^T 1)^{-1}(\eta (Z_1 + Z_2 + 1^T 1) - \Delta_1 - \Delta_2 - D - 1 \delta_3) \tag{14}
\]
Algorithm 1 The implementation of SDLA

**Input:** $X$, $\lambda_1$, $\lambda_2$, $\rho$, $\alpha$. **Initial.:** $k \leftarrow 0$, $D^{(k)}$.
1: repeat
2: Initial. $t \leftarrow 0$, $Z(t) =$ choose the same dissimilarity metric with DS3.
3: while not converged do
4: Update $Z^{(t+1)}$ according to (14);
5: Update $Z_1^{(t+1)}$ according to (15);
6: Update $Z_2^{(t+1)}$ according to (16);
7: $\Delta_1^{(t+1)} \leftarrow \Delta_1^{(t)} + \eta (Z^{(t+1)} - Z_1^{(t+1)});
8: $\Delta_2^{(t+1)} \leftarrow \Delta_2^{(t)} + \eta (Z^{(t+1)} - Z_2^{(t+1)});
9: $\delta_3^{(t+1)} \leftarrow \delta_3^{(t)} + \eta (1^T Z^{(t+1)} - 1^T);
10: $\eta^{(t+1)} \leftarrow \eta^{(t)}$;
11: $t \leftarrow t + 1$;
12: end while
13: $Z^{(k)} \leftarrow Z^{(t)}$;
14: Update $\Theta$ in deep architecture (10);
15: $D \leftarrow d_{ij} = \Phi_{sim} (\Phi_{rep} (\Theta; x_1, \Phi_{rep} (\Theta; x_2)));
16: $D^{(k+1)} = D^{(k)} + \alpha D$; /update with memory
17: $k \leftarrow k + 1$;
18: until Convergence criterion satisfied
**Output:** Optimal solution $Z^* = Z^{(k)}$.

where $Z$ in (14) is computed by equating the partial derivative of (13) with respect to $Z$ to zero.

$$Z_1 = \arg \min_{Z_1} \mathcal{L}(Z_1) = \Gamma_1 \eta^{-1} (Z + \frac{1}{\eta} \Delta_1)$$ (15)

$$Z_2 = \arg \min_{Z_2} \mathcal{L}(Z_2) = \max (\mathcal{S}_2 \eta^{-1} (Z + \frac{1}{\eta} \Delta_2), 0)$$ (16)

where $\Gamma$ and $\mathcal{S}$ are singular value soft-thresholding and shrinkage-thresholding operators, respectively. In detail, for any matrix $A$ and parameter $\gamma$, the form of analytic solution for $\mathcal{S}$ as follows.

$$\mathcal{S}_\gamma (A) = \text{sign}(A) \max(|A| - \gamma, 0)$$ (17)

Then, we have the definition of $\Gamma$ as

$$\Gamma_\gamma (A) = U \mathcal{S}_\gamma (A) V^T$$ (18)

where $A = U \Lambda V^T$ is the singular value decomposition.

### 4 Implementation Framework

In summary, Alg. 1 shows the steps of detailed implementation of the SDLA model in (1). The algorithm should not be terminated until the change of objective value is smaller than a pre-defined threshold (10^-1 in our experiments). In addition, we initialize $D$ by Euclidean distance.

### 5 Experimental Results and Analysis

In this section, we evaluate the performance of SDLA for selecting prototypes on several illustrative problems.

### 5.1 Clustering via Prototypes

To examine the performance of our proposed framework, we consider the problem of text clustering using prototypes that act as cluster centers. We compare our SDLA mainly with prototype selection based clustering methods, including AP [Frey and Dueck, 2007], Kmeans [Duda et al., 2012], Kmedoids [Kaufman and Rousseeuw, 1987], Spectral Clustering (SC) [Ng et al., 2002], LSC [Chen and Cai, 2011], N-SHLLR [Yin et al., 2016] and DS3 [Elhamifar et al., 2016]. To further verify the effectiveness of joint learning, we also compare SDLA with Dis-SLDAP, which is our assignment model in (3) with given dissimilarity metric. Note that many selection methods (e.g. SSDS) cannot be used for clustering.

The standard document collections TDT2 [Cai et al., 2005] is used for this task, which consists of 9 groups of experiments corresponding to different cluster numbers. For each given cluster number, 30 tests are conducted on different randomly chosen clusters and the average performance is computed. The metric, clustering accuracy (AC), is used to measure the quality of prototypes. Table 1 shows the results of those methods, which prove that the proposed method works well with clustering task using prototypes. This is due to the fact that SDLA selects the most representative prototypes, thus improving grouping performance.

### 5.2 Evaluation by Classification

To further evaluate the discrimination of selected prototypes, we employ a classifier to compare the classification results when working on prototype set and source set. Here, we choose the 1-Nearest Neighbor (1-NN) classifier since it is parameter free and the results will be easily reproducible [Fan et al., 2017]. Because the optimal number of prototypes is unknown, we compare each prototype selection method, including no selection (All data), random selection of samples (Rand), Kmedoids [Kaufman and Rousseeuw, 1987], AP [Frey and Dueck, 2007], SDS [Cong et al., 2011], SMRS [Elhamifar et al., 2012], DS3 [Elhamifar et al., 2016], SSDS [Wang et al., 2017], Dis-SLDAP and SDLA, with varying numbers of selected prototypes. To this end, we consider the problems of scene categorization and handwriting recognition. The experiments are conducted on the Fifteen Scene Categories dataset [Lazebnik et al., 2006] and the USPS digit dataset [Hull, 1994], which consist of 15 and 10 classes, respectively. We randomly select 80% of images in each class to form the source training set and use the rest for testing. For scene images, we take the 4096-dimensional CNN features [Simonyan and Zisserman, 2014] as input. Then a subset of training set is selected for 1-NN classification, whose averaged results over 6 times of execution with different training set selections are shown in Figure 5. Obviously, SDLA is the most closest to the ideal performance of using all data. This comes from the fact that SDLA effectively removes confusing samples by selecting the most discriminative prototypes.

1. We choose the same dissimilarity metric with DS3.
Table 1: Clustering accuracy (AC) (%) of different methods on TDT2 Corpus.

<table>
<thead>
<tr>
<th># cluster</th>
<th>AP</th>
<th>Kmeans</th>
<th>Kmedoids</th>
<th>SC</th>
<th>LS1</th>
<th>NSHLRR</th>
<th>DS3</th>
<th>Dis-SDLA</th>
<th>SDLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85.18±1.10</td>
<td>57.74±1.39</td>
<td>96.72±0.90</td>
<td>79.34±2.12</td>
<td>65.31±2.09</td>
<td>86.65±1.28</td>
<td>96.72±1.55</td>
<td>95.39±1.95</td>
<td>97.06±0.83</td>
</tr>
<tr>
<td>2</td>
<td>86.40±1.39</td>
<td>67.74±2.80</td>
<td>96.72±2.14</td>
<td>78.11±1.42</td>
<td>94.36±1.48</td>
<td>89.35±2.33</td>
<td>79.45±0.96</td>
<td>96.13±1.15</td>
<td>97.72±1.01</td>
</tr>
<tr>
<td>3</td>
<td>69.45±2.79</td>
<td>64.74±1.95</td>
<td>65.79±1.65</td>
<td>86.65±1.03</td>
<td>71.84±2.84</td>
<td>82.62±1.93</td>
<td>99.47±0.67</td>
<td>97.91±0.75</td>
<td>98.12±0.74</td>
</tr>
<tr>
<td>4</td>
<td>68.30±1.39</td>
<td>45.11±3.65</td>
<td>60.27±2.17</td>
<td>68.59±3.70</td>
<td>71.95±1.03</td>
<td>78.35±1.27</td>
<td>71.80±2.04</td>
<td>72.82±3.09</td>
<td>76.55±2.82</td>
</tr>
<tr>
<td>5</td>
<td>63.50±1.69</td>
<td>66.05±2.31</td>
<td>69.48±3.95</td>
<td>74.73±1.03</td>
<td>87.68±1.43</td>
<td>74.09±2.38</td>
<td>74.93±1.76</td>
<td>73.21±1.79</td>
<td>80.36±1.18</td>
</tr>
<tr>
<td>6</td>
<td>65.89±1.48</td>
<td>56.65±1.60</td>
<td>53.34±3.63</td>
<td>62.05±2.70</td>
<td>65.78±3.75</td>
<td>70.77±2.27</td>
<td>70.65±1.66</td>
<td>70.23±1.65</td>
<td>76.22±1.16</td>
</tr>
<tr>
<td>7</td>
<td>52.37±3.11</td>
<td>68.49±1.49</td>
<td>58.06±3.95</td>
<td>71.66±2.34</td>
<td>70.02±2.58</td>
<td>71.69±1.22</td>
<td>70.92±3.75</td>
<td>71.29±2.24</td>
<td>78.18±1.51</td>
</tr>
<tr>
<td>8</td>
<td>59.78±3.70</td>
<td>68.51±2.89</td>
<td>65.95±2.96</td>
<td>69.30±2.55</td>
<td>69.66±1.14</td>
<td>70.13±1.14</td>
<td>69.45±3.26</td>
<td>69.10±3.84</td>
<td>75.50±2.25</td>
</tr>
<tr>
<td>9</td>
<td>60.11±1.81</td>
<td>52.83±2.34</td>
<td>58.41±2.92</td>
<td>76.81±1.35</td>
<td>72.50±1.19</td>
<td>79.17±1.25</td>
<td>82.51±1.61</td>
<td>81.33±1.47</td>
<td>86.68±1.35</td>
</tr>
<tr>
<td>10</td>
<td>63.50±2.11</td>
<td>60.30±2.35</td>
<td>64.33±2.69</td>
<td>74.10±2.02</td>
<td>74.32±2.06</td>
<td>78.09±1.67</td>
<td>81.43±1.91</td>
<td>80.77±1.99</td>
<td>85.15±1.42</td>
</tr>
</tbody>
</table>

(a) Figure 5: 1-NN classification results of different prototype selection methods on different datasets. (b) USPS digit dataset.

Figure 6: Percentage of outliers among the prototypes selected by different methods as a function of the fraction of outliers.

5.3 Robustness to Outliers

To evaluate the performance of SDLA for rejecting outliers, we form a dataset of 9000 images, of which 1 − θ fraction are randomly selected from the Extended YaleB face database, and the remaining, corresponding to outliers, are random images downloaded from the internet. For θ ∈ {20%, 30%, 40%, 50%}, we run SDLA as well as Kmedoids, AP, SDS, SMRS, DS3, SDS and Dis-SDLA to select roughly 300 prototypes from the dataset. Figure 6 shows the percentage of outliers among the selected prototypes as we increase the number of outliers in the dataset. Obviously, SDLA results in less outliers compared with other methods. To facilitate the parameter tuning, we run Algorithm 1 with the parameter analysis results for several values of \( \lambda_1 = \lambda_0/\nu \), where \( i = 1, 2 \), \( \lambda_0 \) is computed from \( X \) [Elhamifar et al., 2012], and \( \nu \in \{2, 30\} \). Actually, the sensibility of SDLA degenerates into that of Dis-SDLA. Figure 7 presents the parameter analysis results for several values of \( \nu \).

6 Conclusion

In this work, we introduced a SDLA framework to select representative and discriminative prototypes. Promising experimental results show the effectiveness of SDLA.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2016YFB0800404 and the National Natural Science Foundation of China under Grants 61532005, 61332012, and 61572068, and in part by the Fundamental Research Funds for the Central Universities under Grant 2017JJS056.
References


[Cai et al., 2005] Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. TPAMI, 17(12):1624–1637, 2005.


