Generalized Independent Subspace Clustering

Wei Ye*, Samuel Maurus†‡, Nina Hubig‡ and Claudia Plant§
*Ludwig-Maximilians-Universität München, Munich, Germany
†Helmholtz Zentrum München, Munich, Germany
‡Technische Universität München, Munich, Germany
§University of Vienna, Vienna, Austria
ye@dbs.ifi.lmu.de, samuel.maurus@helmholtz-muenchen.de, hubig@in.tum.de, claudia.plant@univie.ac.at

I. Algorithm

Here we detail ISAAC in two parts, namely 1) automating ISA to find the best $\mathbf{S}$, $\mathbf{W}$ and $\boldsymbol{d}$ (Stages 1–6 in Figure 2), and 2) automating the clustering in each subspace (Stage 7 in Figure 2). We provide an implementation in our supplement[1].

Algorithm 1: Parameter-free ISA

Input: Data $\mathbf{X} \in \mathbb{R}^{m \times n}$.
Output: Matrices $\mathbf{W}_b$ and $\mathbf{S}_b$ ($\mathbf{S}_b = \mathbf{W}_b\mathbf{X}$), vector $\boldsymbol{d}_b$.

1 $\boldsymbol{d} \leftarrow (1, \ldots, 1) \in \{1\}^m$; `/* First candidate */`
2 $\boldsymbol{a} \leftarrow (\{X_1\}, \ldots, \{X_m\})$;
3 $(\mathbf{S}, \mathbf{W}) \leftarrow$ **ISA**$(\mathbf{X}, \boldsymbol{d})$; `/* First ISA result */`
4 $c_b \leftarrow L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\})$;
5 $\mathbf{S}_b \leftarrow \mathbf{S}$, $\mathbf{W}_b \leftarrow \mathbf{W}$, $\boldsymbol{d}_b \leftarrow \boldsymbol{d}$; `/* Track best */`
6 **while** $q > 1$ **do** `/*` $q =$ `length(`$\boldsymbol{d}$`) */`
  `/*` $\boldsymbol{c}$ `has` l_c `tuples: each a merge candidate with` C_M `value */`
7   $\boldsymbol{c} \leftarrow ((a_1, a_2, t_1 = C_M(a_1, a_2)),$
8     $\ldots, (a_{q-1}, a_q, t_{l_c} = C_M(a_{q-1}, a_q)))$;
  `/* Abort if nothing to merge */`
9   **if** $\min(t_1, \ldots, t_{l_c}) > 0$ **then** break;
10   $\boldsymbol{c} \leftarrow$ **order_ascending_by_t_value** $(\boldsymbol{c})$; `/* Sort */`
11   $\boldsymbol{d} \leftarrow ()$, $\boldsymbol{a} \leftarrow ()$; `/* New` $\boldsymbol{d}, \boldsymbol{a}$ `candidates */`
12   **for** $k \leftarrow 1$ **to** l_c **take** c_i **as** $(\mathcal{X}_i, \mathcal{X}_j, t_k)$ **and do**
13   **if** $t_k < 0$ **and** $(\mathcal{X}_i \cup \mathcal{X}_j) \cap \left(\bigcup_{\mathcal{X} \in \boldsymbol{a}} \mathcal{X}\right) = \emptyset$ **then**
      `/* Merge` $\mathcal{X}_i$ `and` $\mathcal{X}_j$ `*/`
14       $\boldsymbol{a}$.**push** $(\mathcal{X}_i \cup \mathcal{X}_j)$, $\boldsymbol{d}$.**push** $(|\mathcal{X}_i \cup \mathcal{X}_j|)$;
15   **else**
16       $\boldsymbol{a}$.**pushAll** $(\mathcal{X}_i, \mathcal{X}_j)$, $\boldsymbol{d}$.**pushAll** $(|\mathcal{X}_i|, |\mathcal{X}_j|)$;
17   $(\mathbf{S}, \mathbf{W}) \leftarrow$ **ISA**$(\mathbf{X}, \boldsymbol{d})$; `/* ISA result */`
18   **if** $L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\}) < c_b$ **then**
19     $c_b \leftarrow L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\})$;
20     $\mathbf{S}_b \leftarrow \mathbf{S}$, $\mathbf{W}_b \leftarrow \mathbf{W}$, $\boldsymbol{d}_b \leftarrow \boldsymbol{d}$;

A. Convergence and Complexity

We first consider the complexity of a single call to ISA (lines 3 and 19). As discussed in Section 2.2, we choose with [2] an ISA implementation which supports heterogeneous subspace dimensionalities. It relies on the *ISA separation principle*, which proposes that the ISA task can be solved by ICA preprocessing and subsequent clustering of the ICA components into statistically-independent groups. The ICA implementation (FastICA) has guaranteed convergence and a worst-case runtime in $\mathcal{O}(nm)$ (assuming its iteration count to be bounded). Given the ICA result, ISA proceeds to group the components into subspaces. This grouping is equivalent to multiplying the ICA mixing matrix $\mathbf{W}$ by a permutation matrix, for which there quickly become an intractable number of possibilities for large m [2]. The implementation hence uses a greedy approach for finding an optimal permutation matrix: it iterates over all pairs of components between subspaces (the count of which is in $\mathcal{O}(m^2)$ for our initial $\boldsymbol{d}$ vector), swapping them when beneficial. It does this for a maximum fixed number of iterations, thus has guaranteed convergence with a worst-case run-time complexity in $\mathcal{O}(nm^2)$. Whitening data, a preprocessing step in ISA, likewise has complexity in $\mathcal{O}(nm^2)$, so the overall run-time complexity of a call to ISA is in $\mathcal{O}(nm^2)$.

Next, we consider the evaluation of coding cost L_I for an ISA solution (lines 4 and 20). For Kernel Density Estimation we use the tractable solution discussed in [1] with time complexity in $\mathcal{O}(nm)$, avoiding the naïve approach's quadratic cost in n. After we have the KDE estimate, equation (3.10) is evaluated in $\mathcal{O}(nm)$ time. The run-time growth rate for evaluating L_I is hence in $\mathcal{O}(nm)$.

On line 7 we compute dependency indicators for each pair of subspaces. The pathological case here is for a candidate $\boldsymbol{d} = (\sqrt{m}, \ldots, \sqrt{m}) \in \mathbb{Z}_+^{\sqrt{m}}$, which implies $\mathcal{O}(m)$ pairs for which we need to calculate the measure C_M. For each combination we again depend on KDE, requiring $\mathcal{O}(n\sqrt{m})$ for each subspace. The run-time of line 7 hence grows with $\mathcal{O}(nm\sqrt{m})$ in the worst case.

Finally, from Section 3.1 we know that the main loop (line 6) has guaranteed worst-case convergence in m iterations (the pathological case for the number of $\boldsymbol{d}$-vector candidates). Algorithm 1 hence has a worst-case run-time complexity in $\mathcal{O}(m\left(nm\sqrt{m} + nm^2 + nm\right)) = \mathcal{O}(nm^3)$.

For a given subspace and k value, clustering with EM_h has a run-time in $\mathcal{O}(nmk)$ (again assuming a bounded number of E-M iterations). Our search for the optimal k for a given subspace introduces an additional loop with worst-case n iterations (again a pathological case; practically the number of iterations is around a few dozen). In the worst-case we also have m subspaces in which to perform clustering, so the worst-case computational complexity of the clustering stage is in $\mathcal{O}(n^2m^2k)$. Assuming the worst-case for both stages we find the **worst-case ISAAC run-time complexity** as

[1] https://github.com/yeweiysh/ISAAC

$\mathcal{O}(nm^3 + n^2m^2k)$.

References

[1] A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In *SDM*, pages 203–211. SIAM, 2003.

[2] Z. Szabó, B. Póczos, and A. Lőrincz. Separation theorem for independent subspace analysis and its consequences. *Pattern Recognition*, 45(4):1782–1791, 2012.