Order Learning Using Partially Ordered Data via Chainization

Seon-Ho Lee[0000−0002−3844−7081] and Chang-Su Kim[0000−0002−4276−1831]

School of Electrical Engineering, Korea University, Korea
seonholee@mcl.korea.ac.kr, changsukim@korea.ac.kr

Abstract. We propose the chainization algorithm for effective order learning when only partially ordered data are available. First, we develop a binary comparator to predict missing ordering relations between instances. Then, by extending the Kahn's algorithm, we form a chain representing a linear ordering of instances. We fine-tune the comparator over pseudo pairs, which are sampled from the chain, and then re-estimate the linear ordering alternately. As a result, we obtain a more reliable comparator and a more meaningful linear ordering. Experimental results show that the proposed algorithm yields excellent rank estimation performances under various weak supervision scenarios, including semi-supervised learning, domain adaptation, and bipartite cases. The source codes are available at https://github.com/seon92/Chainization

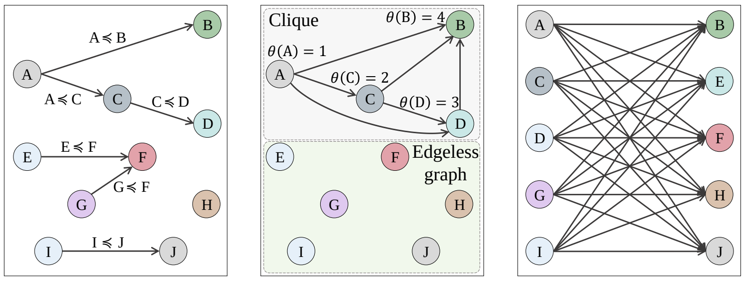
Keywords: Order learning, topological sorting, rank estimation, facial age estimation, aesthetic assessment, facial expression recognition

1 Introduction

In ordered data, objects are sorted according to their classes representing ranks or priorities. For instance, in facial age estimation, face photos are sorted according to the ages. Also, in a video streaming platform, videos can be sorted according to user preferences. For these ordered data, various attempts [5,9,11,12,17,24] have been made to estimate the ranks of objects. In particular, order learning algorithms [9, 12, 24] have shown promising rank estimation performances on diverse ordered data recently.

Order learning is based on the idea that relative assessment is easier than absolute assessment; telling the older one between two people is easier than estimating their exact ages. Hence, in order learning [9, 12], a pairwise comparator to predict pairwise ordering relations is trained. Then, the rank of a test object is estimated by comparing it with the references with known ranks. To obtain a reliable comparator, they exploit the ordering relation for every pair of training objects. This complete ordering information, however, is not always available because it is hard to annotate the exact rank of every object [29].

A partial ordering means that ordering relations are known only for restricted pairs of objects. As illustrated in Fig. 1, we can consider three cases of partial orderings according to the types of underlying graphs.



(a) Random edges (b) Clique-edgeless (c) Bipartite

Fig. 1: Three cases of partial orderings. In each graph, nodes and edges represent objects and known ordering relations, respectively. A directed edge from node x to node y means that $x \preccurlyeq y$. In (b), $\theta(x)$ denotes the rank of x.

- **Random edge case**: Ordering relations are given for randomly selected pairs of objects. For example, an extreme scenario that the relations are known for only 0.01% of all possible pairs is also considered in this paper. This case is represented by a simple directed graph as in Fig. 1(a).
- **Clique-edgeless case**: The ground-truth ranks are available for a subset of training objects. Thus, the ordering relation is known for every pair of objects in the subset, while no information is available for the other pairs. This case is represented by a sum of a clique and an edgeless graph as in Fig. 1(b). Note that typical semi-supervised learning [25,28] and unsupervised domain adaptation [13,30] scenarios belong to this case.
- **Bipartite case**: Objects are partitioned into two groups such that every object in one group has a higher rank than every object in the other group. Within each group, no ordering information is available. For example, face photos are simply dichotomized into either young ones or old ones. This case is represented by a complete bipartite graph in Fig. 1(c).

In these partial ordering cases, order learning yields poor results due to insufficient ordering information for training. To address the lack of supervision, many researches [13, 25, 28, 30] have been conducted. Nevertheless, to the best of our knowledge, no algorithm has been proposed for these partially ordered data. Topological sorting algorithms [7, 8, 19, 20] may be exploited to complement the incomplete ordering information. From a known partial ordering, these algorithms estimate a linear ordering, representing the ordering relation for every pair of objects. However, they do not consider the quality of the resultant linear ordering; for example, if it is known that $w \prec y$ and $x \prec z$, they may yield an arbitrary one of $w \prec x \prec y \prec z$ or $x \prec z \prec w \prec y$ or many other possibilities. Thus, the obtained linear ordering may be unreliable or even meaningless.

In this paper, under the three scenarios in Fig. 1, we aim to obtain a meaningful linear ordering from a given partial ordering and to enhance order learning performances. To this end, we propose a unified approach called *chainization*. First, to obtain a meaningful linear ordering, ordering criteria, such as age in face photos, should be recognized from known ordering relations, and unknown ordering relations should be estimated reliably. Hence, we train a pairwise comparator with given ordering relations. Then, using the comparator, we estimate unknown ordering relations and sort all instances to form a linear ordering. Second, we sample pseudo pairs from the obtained linear ordering and fine-tune the comparator using the pseudo pairs iteratively. The proposed algorithm yields a more accurate comparator than conventional order learning techniques [9, 12], since the pseudo pairs make up for insufficient training data. Last, we estimate the rank of an unseen test object by comparing it with references. Experimental results demonstrate that the proposed algorithm provides meaningful ordering results as well as excellent ranking performances under weakly supervised scenarios, including semi-supervised learning, unsupervised domain adaptation, and bipartite cases.

This work has the following major contributions:

- We improve order learning performances on various types of partially ordered data via the proposed chainization.
- The proposed chainization outperforms conventional techniques in both semi-supervised learning and unsupervised domain adaptation tests.
- We achieve competitive rank estimation performances even with a restricted set of training data. Notably, on the Adience dataset [10], we achieve state-of-the-art age estimation results using less than 0.1% of the training pairs.

2 Related Work

2.1 Order Learning

Lim *et al.* [12] first proposed the notion of order learning, which learns ordering relations between objects and determines the rank of an unseen object. It trains a pairwise comparator to categorize the relation between two objects into one of three cases: one object is bigger than, similar to, or smaller than the other. Then, it predicts the rank of a test object by comparing it with references with known ranks. It yields promising results since relative assessment is easier than absolute assessment in general. Based on similar motivations, some learning-to-rank methods [18, 26] also model a ranking function to estimate the priorities of objects via pairwise comparisons. Also, Shin *et al.* [24] modified the classification approach in [12] to develop a regression-based order learning algorithm.

However, not every pair of objects are easily comparable. Hence, Lee and Kim [9] proposed the order-identity decomposition network to decompose object information into an order-related feature and an identity feature. They showed that objects with similar identity features can be compared more reliably. These order learning techniques [9, 12, 24] assume that the rank of every training object is known. In contrast, we assume that only pairwise ordering relations between limited pairs of objects are known. Then, we attempt to discover the ordering relations across all objects from the given incomplete data. As a result, the proposed algorithm can reduce the amount of annotated pairs required for order learning significantly $\left(e.g.\ \text{by a factor of } \frac{1}{100}\right)$.

2.2 Linear Extension of Partial Order

In order theory [6,23], linear extension of a partial order means finding a linear order compatible with the partial order. In other words, an ordering relation between any elements should be determined without conflicting with the partial order. By generating a directed graph for the partial order, this problem can be converted to the topological sorting of the vertices in the graph [3]: linear extension and topological sorting are the same problem.

Various algorithms [7,8,19,20,27] have been proposed for linear extension. In [7], Kahn proposed a simple algorithm based on the breadth-first search (BFS). It first constructs a directed graph, in which vertices correspond to objects and directed edges represent ordering relations. It then repeatedly outputs a vertex with no incoming edge and deletes its outgoing edges from the graph. However, the Kahn's algorithm decides unknown ordering relations arbitrarily. Thus, one may want to obtain all possible linear orders and then evaluate each of them to find the best one. By exploiting the backtracking, Knuth [8] developed an algorithm to generate all possible linear extension results.

Depth-first search (DFS) algorithms also have been proposed. Tarjan [27] and Reingold *et al.* [20] use DFS to obtain a spanning forest of the directed graph, and then output the vertices before any of their descendants in the forest. In [19], DFS is performed both forward and backward to reduce the time complexity. These DFS methods [19,20,27] also determine unknown ordering relations randomly. In contrast, the proposed algorithm yields a meaningful linear ordering by estimating the missing ordering relations.

3 Proposed Algorithm

3.1 Preliminary

Mathematically, an *order* or *partial order* [23] is a binary relation, denoted by $\leq$, on a set $\Theta = \{\theta_1, \theta_2, \ldots, \theta_c\}$ that satisfies the three properties of

- Reflexivity: $\theta_i \leq \theta_i$ for all i;
- Antisymmetry: $\theta_i \leq \theta_j$ and $\theta_j \leq \theta_i$ imply $\theta_i = \theta_j$;
- Transitivity: $\theta_i \leq \theta_j$ and $\theta_j \leq \theta_k$ imply $\theta_i \leq \theta_k$.

Then, Θ is called a *partially ordered set*. Furthermore, if every pair of elements is comparable ($\theta_i \leq \theta_j$ or $\theta_j \leq \theta_i$ for all i, j), Θ is called a *chain* or *linearly ordered set*. In such a case, the partial order is called a *linear order*.

In practice, an order describes the ranks or priorities of classes in the set $\Theta = \{\theta_1, \ldots, \theta_c\}$, where each class represents one or more object instances. For example, in age estimation, θ_i may represent i-year-olds, and $\theta_{20} < \theta_{42}$ represents that 20-year-olds are younger than 42-year-olds. Let $\theta(\cdot)$ be the class function, and let x and y be instances. For example, $\theta(x) = \theta_{20}$ means that person x is 20-year-old. To represent the *ordering* between instances, we use '$\prec, \approx, \succ, \preccurlyeq, \succcurlyeq$' instead of '$<, =, >, \leq, \geq$' to avoid confusion. Specifically, $x \prec y$, $x \approx y$, and $x \preccurlyeq y$ mean that $\theta(x) < \theta(y)$, $\theta(x) = \theta(y)$, and $\theta(x) \leq \theta(y)$, respectively. Also, we use the expression *ordering* to describe instance relations, while using *order* exclusively for class relations.

3.2 Problem Definition

Suppose that there are n training instances in $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. Let $\mathcal{L} = \{(x, y) : x \preccurlyeq y \text{ and } x, y \in \mathcal{X}\}$ be the set of increasingly ordered pairs of instances whose ordering relations are known. In order learning [9, 12, 24], it is assumed that $\mathcal{L}$ is a *linear ordering* of instances:

$$(x, y) \in \mathcal{L} \;\text{ or }\; (y, x) \in \mathcal{L} \;\text{ for all } x, y \in \mathcal{X}. \tag{1}$$

Note that both (x, y) and (y, x) belong to $\mathcal{L}$ if $x \approx y$, and thus $|\mathcal{L}| \geq \binom{n}{2}$. In other words, order learning assumes that the ground-truth class of every training instance is known, as well as the linear order on the set of classes. However, such information may be unavailable. In age estimation, annotating the exact age of a person is difficult, but telling the older one between two people is relatively easy [29]. Therefore, only the binary ordering information (*i.e.* who is older) between some selected pairs of people may be available. In such a case, we are given a *partial ordering* of instances,

$$\mathcal{P} = \{(x, y) : \text{It is known that } x \preccurlyeq y\} \subset \mathcal{L}. \tag{2}$$

We consider the case that the number of ordered pairs in $\mathcal{P}$ is considerably smaller than that in $\mathcal{L}$, $|\mathcal{P}| \ll |\mathcal{L}|$. Then, we formulate the problem as follows.

Problem. *Given a partial ordering $\mathcal{P}$ of instances, the objective is to obtain its superset $\mathcal{L}$ that is a linear ordering.*

In other words, we aim to linearly extend or 'chainize' $\mathcal{P}$ to $\mathcal{L}$. To this end, we propose the chainization algorithm. Note that if $\mathcal{L}$ is estimated reliably, order learning performance can be enhanced by using $\mathcal{L}$ as auxiliary information for training. The chainization algorithm also produces a pairwise comparator, using which we can estimate the rank of an unseen test instance. First, in Section 3.3, we present the chainization algorithm on the random edge case in Fig. 1(a). Then, we describe how to apply the chainization to the other cases in Section 3.4. Last, we explain the rank estimation scheme in Section 3.5.

3.3 Chainization – Basics

The chainization algorithm extends a partial ordering $\mathcal{P}$ on an instance set $\mathcal{X}$ to a linear ordering $\mathcal{L}$. First, we train a pairwise comparator using available information. Second, we use the comparator to estimate the ordering between every pair of instances in $\mathcal{X}$, yielding a linear ordering $\mathcal{L}$. These two steps are iterated to refine both the comparator and the linear ordering.

Graph representation of partial ordering: We use a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$ to represent a partial ordering $\mathcal{P}$ of instances in $\mathcal{X}$. Initially, we

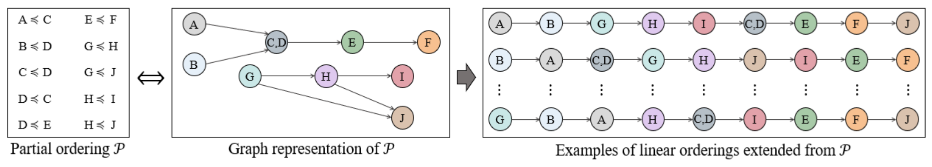

A ≼ C
E ≼ F
B ≼ D
G ≼ H
C ≼ D
G ≼ J
D ≼ C
H ≼ I
D ≼ E
H ≼ J
Partial ordering $\mathcal{P}$
⟺
Graph representation of $\mathcal{P}$
Examples of linear orderings extended from $\mathcal{P}$

Fig. 2: Graph representation of a partial ordering $\mathcal{P}$ and its possible linear extension results.

construct the vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ so that each vertex v_i corresponds to an instance $x_i \in \mathcal{X}$. We also construct the edge set $\mathcal{E} = \{(v_i, v_j) : (x_i, x_j) \in \mathcal{P}\}$ so that there is a directed edge from v_i to v_j if $x_i \preccurlyeq x_j$. In the initial graph, cycles may occur because both (x_i, x_j) and (x_j, x_i) may belong to $\mathcal{P}$ if $x_i \approx x_j$. In such a case, we merge all vertices on each cycle into a single vertex and modify incident edges accordingly. Consequently, each vertex represents a set of one or more instances, which equal ($\approx$) one another.

Fig. 2 shows an example of the graph representation: the partial ordering $\mathcal{P}$ is defined on a set $\mathcal{X}$ of 10 instances, but there are 9 vertices only because 'C' and 'D' are merged into one vertex.

After constructing the graph G, the linear extension of $\mathcal{P}$ to $\mathcal{L}$ can be regarded as finding a vertex sorting function

$$\sigma : \mathcal{V} \to \{1, 2, \ldots, |\mathcal{V}|\} \tag{3}$$

satisfying the constraint

$$\sigma(v_i) < \sigma(v_j) \ \text{ for all } \ (v_i, v_j) \in \mathcal{E}. \tag{4}$$

Note that $\sigma(\cdot)$ is a sorting index. For example, $\sigma(v_i) = 1$ means that v_i is the first in the sorted list of all vertices. If σ is obtained, a linear ordering $\mathcal{L}$ can be easily derived from the σ;

$$\mathcal{L} = \{(x_i, x_j) : \sigma(v_i) \leq \sigma(v_j) \text{ and } x_i, x_j \in \mathcal{X}\} \tag{5}$$

where v_i and v_j are the vertices containing x_i and x_j, respectively. It is guaranteed that $\mathcal{L} \supset \mathcal{P}$ due to the constraint in (4), but a linearly extended ordering $\mathcal{L}$ is not unique in general. As in Fig. 2, there are many possible linear orderings extended from the same partial ordering $\mathcal{P}$. Among them, we aim to determine a desirable linear ordering, which sorts all instances in $\mathcal{X}$ in a meaningful way.

Comparator: To obtain such an ordering, we develop a pairwise comparator in Fig. 3, which classifies the ordering between instances x and y into two cases: $x \preccurlyeq y$ or $x \succcurlyeq y$. The Siamese feature extractor [2] maps x and y to feature vectors, respectively, and then the classifier yields a softmax probability $p^{xy} = (p^{xy}_{\preccurlyeq}, p^{xy}_{\succcurlyeq})$. We first train the comparator using the known ordered pairs in $\mathcal{P}$. Specifically, we optimize it to minimize the loss

$$\ell = [x \not\approx y]\ell_{\mathrm{ce}}(p^{xy}, q^{xy}) + [x \approx y]D(p^{xy} \| q^{xy}) \tag{6}$$

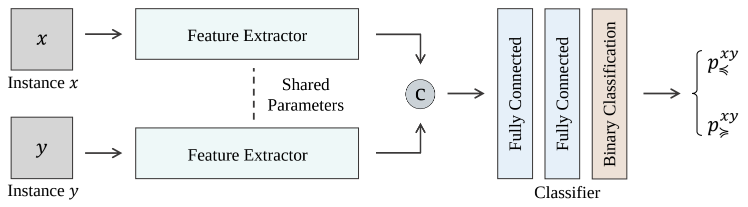


Fig. 3: An overview of the pairwise comparator, where ⓒ denotes concatenation.

Algorithm 1 Chainization

```
Input: Directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$ for $\mathcal{P}$
 1: Train a comparator on $\mathcal{P}$ for warm-up epochs;
 2: repeat
 3:     $\mathcal{Q} \leftarrow \varnothing$;    $t \leftarrow 1$;
 4:     Add all vertices $v \in \mathcal{V}$ with $\delta(v) = 0$ to $\mathcal{Q}$;
 5:     while $\mathcal{Q} \neq \varnothing$ do
 6:         Remove the optimal $v^*$ in (8) from $\mathcal{Q}$;
 7:         $\sigma(v^*) \leftarrow t$;    $t \leftarrow t + 1$;
 8:         for all adjacent vertex $w$ of $v^*$ in $G$ do
 9:             Remove edge $(v^*, w)$ from $\mathcal{E}$;
10:             if $\delta(w) = 0$ then
11:                 Add $w$ to $\mathcal{Q}$;
12:             end if
13:         end for
14:     end while
15:     Obtain a chain from the sorting function $\sigma$;
16:     Shorten it to yield the linear ordering $\mathcal{L}$ in (9);
17:     Build a set $\mathcal{T}$ of pseudo pairs;
18:     Fine-tune the comparator on $\mathcal{P} \cup \mathcal{T}$;
19: until predefined number of epochs;
Output: Linear ordering $\mathcal{L}$, comparator
```

where $[\cdot]$ is the indicator function. If $x \prec y$ or $x \succ y$, we use the cross-entropy loss ℓ_{ce} with the ground-truth one-hot vector $q^{xy} = (q^{xy}_{\preccurlyeq}, q^{xy}_{\succcurlyeq})$. However, if $x \approx y$, we set $q^{xy}_{\preccurlyeq} = q^{xy}_{\succcurlyeq} = 0.5$ and use the KL-divergence D, instead of the cross-entropy. This is because the cross-entropy loss with $q^{xy} = (0.5, 0.5)$ produces near zero gradients for most p^{xy}, delaying the training unnecessarily.

Chainization: To determine the sorting function σ in (3), equivalently to find the linear ordering $\mathcal{L}$ in (5), we propose the chainization algorithm in Algorithm 1, which is based on the Kahn's topological sorting algorithm [7]. However, whereas the Kahn's algorithm obtains an arbitrary linear extension of $\mathcal{P}$, the proposed algorithm yields a meaningful linear ordering by estimating missing ordering information, not included in $\mathcal{P}$, using the pairwise comparator.

As in [7], we iteratively select a vertex v from the graph G and append it to the sorted list. In other words, at iteration t, we select v and set $\sigma(v) = t$. First,

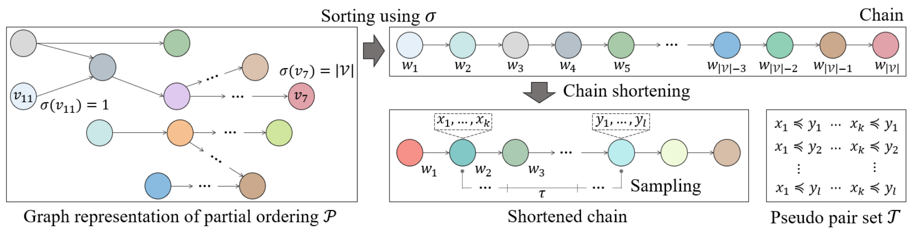

Sorting using σ
Chain
$\sigma(v_7) = |\mathcal{V}|$
v_{11} $\sigma(v_{11}) = 1$
v_7
w_1 w_2 w_3 w_4 w_5 $w_{|\mathcal{V}|-3}$ $w_{|\mathcal{V}|-2}$ $w_{|\mathcal{V}|-1}$ $w_{|\mathcal{V}|}$
Chain shortening
$x_1, \dots, x_k$
$y_1, \dots, y_l$
Sampling
τ
$x_1 \preccurlyeq y_1 \cdots x_k \preccurlyeq y_1$
$x_1 \preccurlyeq y_2 \cdots x_k \preccurlyeq y_2$
$x_1 \preccurlyeq y_l \cdots x_k \preccurlyeq y_l$
Graph representation of partial ordering $\mathcal{P}$
Shortened chain
Pseudo pair set $\mathcal{T}$

Fig. 4: An overview of the chainization.

we form a set $\mathcal{Q}$ to include all vertices v with indegree $\delta(v) = 0$. Note that at least one such vertex with no incoming edge exists, because the graph G for $\mathcal{P}$ is acyclic by construction. Also, such vertices precede the others in G. Second, we select an optimal vertex v^* from $\mathcal{Q}$, which is most likely to contain the smallest instances (*e.g.* the youngest people in age estimation). To this end, we define the probability that a vertex v precedes another vertex w as

$$p(v, w) = \tfrac{1}{kl} \textstyle\sum_{i=1}^{k} \sum_{j=1}^{l} p_{\preccurlyeq}^{x_i y_j} \tag{7}$$

where $v = \{x_1, \dots, x_k\}$ and $w = \{y_1, \dots, y_l\}$. We also define the priority score π of each vertex $v \in \mathcal{Q}$ as $\pi(v) = \sum_{w \in \mathcal{Q}: w \neq v} p(v, w)$. We then choose the highest priority vertex

$$v^* = \arg\max_{v \in \mathcal{Q}} \pi(v) \tag{8}$$

and set $\sigma(v^*) = t$. Then, we remove all outgoing edges of v^* from $\mathcal{E}$. We repeat this process until $\mathcal{Q} = \varnothing$ and thus $\sigma(v)$ is determined for all $v \in \mathcal{V}$.

Pseudo pair sampling: The sorting function σ lists all vertices in $\mathcal{V}$ increasingly, which can be represented by a chain as illustrated in Fig. 4. Let $(w_1, w_2, \dots, w_{|\mathcal{V}|})$ denote this chain, where $w_i = v_j$ if $\sigma(v_j) = i$. The chain may represent linearly ordered classes. However, in general, $|\mathcal{V}|$ is much larger than the number of actual classes, since the graph representation is performed without full annotations of instance equalities ($\approx$).

We hence merge vertices in the chain, which likely come from the same underlying class. Specifically, we merge the adjacent vertices w_i and w_{i+1} with the lowest probability $p(w_i, w_{i+1})$ in (7) into one vertex. This is because a low $p(w_i, w_{i+1})$ implies that the instances in w_i are not clearly smaller ($\prec$) than those in w_{i+1}, and all those instances may belong to the same class. However, for any $x \in w_i$ and $y \in w_{i+1}$, if $x \prec y$ is known in the partial ordering $\mathcal{P}$, the merging is not allowed and the pair with the second lowest probability is merged. This is performed iteratively, until the number of vertices reaches a predefined threshold or no vertices can be merged, to yield a shortened chain.

The linear ordering $\mathcal{L}$ can be derived from the shortened chain by

$$\mathcal{L} = \{(x, y) : x \in w_i, y \in w_j \text{ and } i \leq j\}. \tag{9}$$

Notice that $\mathcal{L}$ is obtained using the output of the comparator in (7), which is trained on the partial ordering $\mathcal{P}$. The additional information in $\mathcal{L}$, in turn, can be used to fine-tune the comparator. To this end, as shown in Fig. 4, we form a set $\mathcal{T}$ of pseudo training pairs. First, we sample an ordered pair (x, y), where $x \in w_i$, $y \in w_j$, and $j - i > \tau$, and add it to $\mathcal{T}$. It is called a *pseudo* pair, since the ordering $x \prec y$ is an estimated result using the comparator, instead of a ground-truth in the training set $\mathcal{P}$. In general, a larger threshold τ yields a more reliable $\mathcal{T}$. However, at a large τ, sampled pairs may be less informative, for their relations are relatively easy to predict. The impacts of τ will be analyzed in the supplemental document. Second, we also sample every possible pair (x, y) from each vertex w and add both (x, y) and (y, x) to $\mathcal{T}$ to indicate that $x \approx y$.

Next, we fine-tune the comparator using the augmented training set $\mathcal{P} \cup \mathcal{T}$. The comparator is, in turn, used to update the linear ordering $\mathcal{L}$ and the pseudo pair set $\mathcal{T}$. This is repeated as described in Algorithm 1.

3.4 Chainization – Applications

Clique-edgeless case: Let us consider the clique-edgeless case in which training instances in $\mathcal{X}$ are partitioned into two subsets $\mathcal{X}_\mathrm{c}$ and $\mathcal{X}_\mathrm{e}$, and the ground-truth ranks are available only for the instances in $\mathcal{X}_\mathrm{c}$. Thus, the instances in $\mathcal{X}_\mathrm{c}$ form a clique, whereas those in $\mathcal{X}_\mathrm{e}$ form an edgeless subgraph in G, as shown in Fig. 1(b). Note that the clique-edgeless case can represent the semi-supervised learning scenario [25, 28] if $|\mathcal{X}_\mathrm{c}| \ll |\mathcal{X}_\mathrm{e}|$. Also, it can represent the unsupervised domain adaptation scenario [13, 30] when $\mathcal{X}_\mathrm{c}$ and $\mathcal{X}_\mathrm{e}$ are different source and target datasets, respectively.

The proposed chainization can be applied to this clique-edgeless case as well. We first train the comparator using the known ordering relations on $\mathcal{X}_\mathrm{c}$. Then, we sort all instances via the chainization. However, at early iterations, the instances in $\mathcal{X}_\mathrm{e}$ may not be ordered reliably, since no supervision is provided for them. Hence, we do not shorten the chain at early iterations. Also, to form a pseudo pair set $\mathcal{T}$, we use a gradually decreasing threshold τ, so that we can sample more reliable pairs at early iterations and more informative pairs at later iterations. The supplemental document describes the scheduling of τ in detail.

Bipartite case: Training instances in $\mathcal{X}$ are partitioned into two subsets $\mathcal{X}_0$ and $\mathcal{X}_1$, and the only annotations are that every instance in $\mathcal{X}_0$ is no larger than ($\preccurlyeq$) every instance in $\mathcal{X}_1$. This special partial ordering $\mathcal{P}$ is represented by a complete bipartite graph G in Fig. 1(c). Even in this challenging case, the chainization algorithm can sort all instances in $\mathcal{X}$ meaningfully and yield a chain by adopting the same strategy used in the clique-edgeless case.

3.5 Rank Estimation

Based on order learning [12], we can estimate the rank of an unseen instance x by comparing it with multiple references with known ranks. For the rank estimation, Lee and Kim [9] developed the MAP estimator. However, their algorithm adopts a ternary classifier as the comparator and yields a probability vector of $p^{xy} = (p^{xy}_{\prec}, p^{xy}_{\approx}, p^{xy}_{\succ})$ by comparing x with reference y. In contrast, we do not compute the probability $p^{xy}_{\approx}$ of the equal case explicitly, since it requires a threshold to define the equality between instances [12]. Thus, we modify the MAP estimation rule accordingly, which is detailed in the supplemental document.

4 Experiments

We conduct various experiments on facial age estimation [10, 21], aesthetic assessment [22] and facial expression recognition [1] datasets to assess the proposed algorithm under the three different scenarios: random edge case, clique-edgeless case, and bipartite case. Due to the space limitation, implementation details and more results are available in the supplemental document.

4.1 Datasets

MORPH II [21]: It provides 55,134 facial images labeled with the exact ages in range $[16, 77]$. For evaluation, we select 5,492 images of the Caucasian race and divide them randomly into two subsets: 80% for training and 20% for test.

Adience [10]: It contains 26,580 images annotated with one of the eight age group labels: '0–2,' '4–6,' '8–13,' '15–20,' '25–32,' '38–43,' '48–53,' and '60–100.' For evaluation, we adopt the standard 5-fold cross validation [5, 10, 11].

Aesthetics [22]: It provides 15,687 image URLs on Flickr, where 13,929 images are available but the others are lost. Each image is annotated with a 5-scale aesthetic score. We use the 5-fold cross validation for evaluation.

FER+ [1]: It contains 32,298 grayscale images for facial expression recognition. Each image is categorized by 10 annotators into one of eight emotion classes, and the ground-truth class is determined by the majority rule.

4.2 Metrics

Linear extension: To measure the quality of linear extension of $\mathcal{P}$ to $\mathcal{L}$, we use two metrics: Spearman's ρ [4] and pairwise error (PE). The Spearman's ρ computes the correlation coefficient between two instance rankings, which correspond to an estimated linear ordering $\hat{\mathcal{L}}$ and its ground-truth $\mathcal{L}$, respectively. PE is defined as $\mathrm{PE} = 1 - 1/|\hat{\mathcal{L}}| \sum_{(x,y)\in\hat{\mathcal{L}}}[(x, y) \in \mathcal{L}]$, which measures the ratio of disordered pairs in $\hat{\mathcal{L}}$.

Rank estimation: We assess rank estimation results by the mean absolute error (MAE) and the classification accuracy. MAE is the average absolute error between estimated and ground-truth ranks. For the classification accuracy, the closest rank to the MAP estimation result is regarded as the estimated rank. Note that rank estimation can be regarded as finding a meaningful linear ordering of unseen test instances.

Table 1: Linear extension results on MORPH II, Adience, and Aesthetics.

	MORPH II						Adience						Aesthetics					
	$\gamma = 0.05\%$		$\gamma = 0.1\%$		$\gamma = 0.15\%$		$\gamma = 0.01\%$		$\gamma = 0.02\%$		$\gamma = 0.03\%$		$\gamma = 0.01\%$		$\gamma = 0.02\%$		$\gamma = 0.03\%$	
	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)	PE ($\downarrow$)	ρ ($\uparrow$)
Lower Bounds [7]	0.419	0.205	0.321	0.483	0.238	0.698	0.407	0.057	0.353	0.225	0.281	0.439	0.270	0.057	0.274	0.106	0.290	0.119
DRA [26]	0.126	0.892	0.108	0.924	0.099	0.927	0.115	0.814	0.113	0.823	0.095	0.851	0.080	0.679	0.071	0.802	0.066	0.836
OL [12]	0.128	0.897	0.108	0.923	0.096	0.930	0.120	0.809	0.113	0.821	0.100	0.847	0.083	0.665	0.069	0.805	0.067	0.822
Proposed	**0.114**	**0.918**	**0.100**	**0.936**	**0.089**	**0.949**	**0.061**	**0.908**	**0.033**	**0.948**	**0.027**	**0.959**	**0.063**	**0.838**	**0.052**	**0.853**	**0.034**	**0.872**

4.3 Random Edge Case

First, we evaluate the linear extension and rank estimation performances in the random edge case. Let $\gamma = 100 \times \frac{|\mathcal{P}|}{|\mathcal{L}|}$ denote the percentage of available known pairs in a partial ordering $\mathcal{P}$ over all pairs in the linear ordering $\mathcal{L}$. Hence, at a lower γ, the linear extension of $\mathcal{P}$ to $\mathcal{L}$ is more difficult.

Linear extension: Table 1 summarizes the linear extension results on three datasets: MORPH II, Adience, and Aesthetics. Note that there is no conventional technique to extend a partial ordering to a meaningful linear ordering. Hence, for comparison, we provide the results of the Kahn's topological sorting algorithm [7] and conventional comparison-based relative rank estimators, DRA [26] and OL [12], which also can be trained on a partial ordering $\mathcal{P}$. The Kahn's algorithm yields one of the possible linear orderings arbitrarily, so it performs poorly in terms of both PE and Spearman's ρ. Nevertheless, its results can be regarded as the performance lower bounds. Both DRA and OL do not provide a method to extend $\mathcal{P}$ to $\mathcal{L}$. Hence, instances are sorted by their estimated ranks.

First, even at $\gamma = 0.05\%$ on MORPH II, the proposed algorithm achieves a high ρ of 0.918 using only 0.05% of ordered pairs in $\mathcal{L}$ as supervision. This indicates that the proposed algorithm predicts the other missing 99.95% pairs in $\mathcal{L}$ reliably. Second, Adience has eight age group classes, whereas MORPH II has more than 60 age classes. Hence, the linear ordering $\mathcal{L}$ of the Adience data can be more easily estimated: even when γ is as low as 0.01%, the proposed algorithm obtains a high ρ of 0.908 and a low PE of 0.061. Third, due to subjectivity and ambiguity of aesthetic criteria, Aesthetics is more challenging than Adience is. So, it yields relatively low scores at the same γ.

Rank estimation: Fig. 5 compares the rank estimation (age group classification) accuracies of the proposed algorithm and the conventional order learning algorithm OL [12] on the Adience test set. To estimate the rank of an instance, OL requires reference instances with known ranks because it performs comparison-based rank estimation. Thus, for each rank, an instance is randomly selected from the training set as a reference. The proposed algorithm also uses the same references for the rank estimation. At all γ's, the proposed algorithm outperforms OL. Especially, at a low $\gamma = 0.005\%$, OL fails to obtain a reliable comparator due to the lack of training pairs, resulting in a poor accuracy of 31.8%. In contrast, the proposed algorithm achieves a much higher accuracy of 56.7% by optimizing the comparator over pseudo pairs.

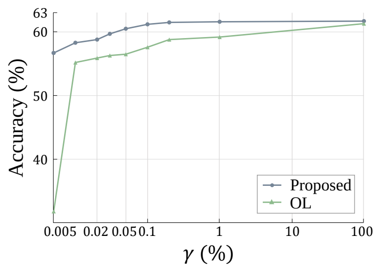


Fig. 5: Comparison of the proposed chainization with OL [12] on Adience. The x-axis is in a logarithmic scale.

Table 2: Comparison of rank estimation results on Adience.

Algorithm	Accuracy (%)	MAE
OR-CNN [16]	56.7 ± 6.0	0.54 ± 0.08
CNNPOR [14]	57.4 ± 5.8	0.55 ± 0.08
GP-DNNOR [15]	57.4 ± 5.5	0.54 ± 0.07
SORD [5]	59.6 ± 3.6	0.49 ± 0.05
POE [11]	60.5 ± 4.4	0.47 ± 0.06
Proposed ($\gamma = 100\%$)	$\mathbf{61.7 \pm 4.3}$	$\mathbf{0.46 \pm 0.05}$
Proposed ($\gamma = 0.08\%$)	60.5 ± 4.2	0.48 ± 0.05
Proposed ($\gamma = 0.03\%$)	59.7 ± 4.0	0.49 ± 0.05
Proposed ($\gamma = 0.02\%$)	58.8 ± 4.2	0.51 ± 0.06
Proposed ($\gamma = 0.01\%$)	58.3 ± 4.5	0.53 ± 0.06

Table 2 compares the proposed algorithm with conventional ordinal regressors [5, 11, 14–16] on the Adience test set. We provide the results of the proposed algorithm at $\gamma = 100\%$ as the performance upper bounds, which achieve the best scores among all methods. Here, the comparator is trained using the ground-truth linear ordering $\mathcal{L}$ of the training set. With weaker supervision, the performances are lowered but still competitive. For example, using only 0.03% of the ordering relations in $\mathcal{L}$, the proposed algorithm performs better than the others, except for POE [11]. Moreover, at $\gamma = 0.08\%$, the proposed algorithm reaches the performances of POE. This confirms that the comparator is effectively fine-tuned with the partial data augmented by pseudo pairs.

4.4 Clique-Edgeless Case

To assess the proposed algorithm in the clique-edgeless case, we employ typical semi-supervised learning and unsupervised domain adaptation protocols.

First, in the semi-supervised learning test, we compare the proposed algorithm with the state-of-the-art ordinal regressors, POE [11] and SORD [5], at various supervision levels s, which means that the ground-truth ranks are known for s% of the training instances. However, the ordinal regressors assume that the ground-truth rank of every training instance is given. Hence, we utilize recent semi-supervised learning algorithms, FlexMatch [28] and FixMatch [25], together with the ordinal regressors so that they can use unlabeled instances for training. Fig. 6 compares the results. The performances of SORD are severely degraded at low levels of s. Its performances do not improve even when it is combined with FlexMatch and FixMatch. Compared to SORD, POE and its combined versions with FlexMatch and FixMatch provide better results. However, the proposed algorithm achieves the best accuracies at all levels of s with large margins. Notably, the proposed algorithm at $s = 30\%$ shows competitive results to the fully supervised POE. This indicates that the chainization can effectively reduce the amount of supervision required for obtaining a good rank estimator.

Next, we compare the proposed algorithm under the domain adaptation protocol. We use Adience and MORPH II as the source and target domains, respectively. Table 3 compares the sorting and rank estimation results. To compute ρ

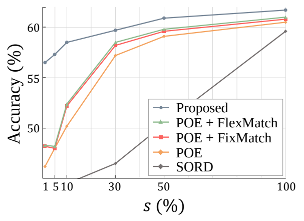


Fig. 6: Semi-supervised learning performances on Adience.

Table 3: Domain adaptation results from Adience to MORPH II.

Algorithm	ρ	MAE	Accuracy (%)
POE [11]	0.614	0.66	43.4
SORD [5]	0.560	0.85	37.5
OL [12]	0.630	0.68	43.8
Proposed	**0.798**	**0.54**	**51.5**

of POE and SORD, instances are sorted by their estimated ranks. Although the source and target datasets contain images of different characteristics, the proposed algorithm performs reliably and provides better results than the conventional algorithms in [5,11,12]. Notably, compared to OL, the proposed algorithm improves the performances meaningfully via the chainization.

4.5 Bipartite Case

In the bipartite case, all instances in $\mathcal{X}$ are partitioned into two subsets $\mathcal{X}_0$ and $\mathcal{X}_1$ so that every instance in $\mathcal{X}_0$ is smaller than every instance in $\mathcal{X}_1$. In other words, every ordering relation across the two subsets is known. Hence, to evaluate the quality of linear extension, we measure the ρ and PE scores for $\mathcal{X}_0$ and $\mathcal{X}_1$ separately and report the average scores over the subsets.

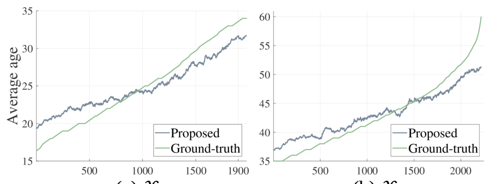
Table 4 lists the performances on MORPH II and Adience. The Kahn's algorithm [7] yields almost zero ρ's, since no ordering information within $\mathcal{X}_0$ or $\mathcal{X}_1$ is available in $\mathcal{P}$. In contrast, the proposed algorithm sorts the instances in $\mathcal{X}$ meaningfully, outperforming DRA and OL by large margins. The ρ coefficient for Adience is relatively low since its subset $\mathcal{X}_1$ has a severe class imbalance. Also, we compare the rank estimation results of OL and the proposed algorithm in terms of MAE. In this test, we use randomly selected references for each rank. Due to the extremely limited information for training, OL yields poor results. In contrast, the proposed algorithm provides decent MAE results, 5.9 for the 62 ranks in MORPH II and 0.8 for the 8 ranks in Adience.

Fig. 7 shows the linear extension results on MORPH II in more detail. After sorting the instances in $\mathcal{X}$ based on $\hat{\mathcal{L}}$, we compute the moving average age of 100 consecutive instances from the youngest to the oldest. Although no ordering information within $\mathcal{X}_0$ or $\mathcal{X}_1$ is available, the proposed algorithm estimates such information quite reliably and yields generally increasing curves.

Last, we assess the performances on the FER+ dataset [1]. There is no explicit order between the emotion classes in FER+, so we consider the two classes of 'sadness' and 'happiness' and assume that 'sadness' precedes ($<$) 'happiness.' In other words, 'sadness' is assumed to be the opposite feeling of 'happiness' on the same axis. Then, the instances belonging to 'sadness' and 'happiness' are assigned to $\mathcal{X}_0$ and $\mathcal{X}_1$, respectively. Even in this challenging case, the chainiza-

Table 4: Linear extension results in the bipartite case.

	MORPH II			Adience		
	MAE(↓)	PE (↓)	ρ (↑)	MAE(↓)	PE (↓)	ρ (↑)
Lower Bounds [7]	-	0.484	0.003	-	0.341	0.007
DRA [26]	-	0.334	0.405	-	0.270	0.228
OL [12]	8.696	0.450	0.068	1.682	0.307	0.128
Proposed	**5.903**	**0.246**	**0.638**	**0.807**	**0.192**	**0.452**



(a) $\mathcal{X}_0$ (b) $\mathcal{X}_1$

Fig. 7: Sorting $\mathcal{X}_0$ and $\mathcal{X}_1$ of MORPH II.

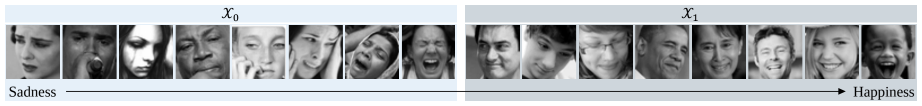


Fig. 8: Sorting of the instances in 'sadness' and 'happiness' classes in the FER+ dataset. More examples are shown in the supplemental document.

tion can sort all the instances in a meaningful order. Fig. 8 shows examples of the sorted instances. The instances in the 'sadness' class are sorted meaningfully from 'weeping' to 'wailing,' and those in the 'happiness' class are from 'smiling tightly' to 'laughing.' The proposed algorithm discovers these subclasses without any intra-class supervision; it discovers them based on the inter-class order assumption only. The proposed algorithm determines that the wailing instance is happier than the weeping one. One possible explanation for this counter-intuitive ordering is that wailing entails emitting a cry by opening one's mouth wide, similar to when one laughs.

5 Conclusions

We proposed the chainization algorithm to improve order learning performances on partially ordered data. First, we estimate unknown ordering relations of instances using a comparator trained on the partial ordering. Then, based on the estimated relations, we obtain a linear ordering and then sample pseudo pairs. We then fine-tune the comparator using the pseudo pairs iteratively. As a result, we obtain a more accurate comparator than conventional order learning. Extensive experiments on various datasets showed that the proposed algorithm provides meaningful sorting results and excellent rank estimation performances under diverse weak supervision scenarios.

Acknowledgments

This work was supported by the NRF grants funded by the Korea government (MSIT) (No. NRF-2021R1A4A1031864 and No. NRF-2022R1A2B5B03002310) and also by IITP grant funded by the Korea government (MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub).

References

1. Barsoum, E., Zhang, C., Ferrer, C.C., Zhang, Z.: Training deep networks for facial expression recognition with crowd-sourced label distribution. In: ICMI (2016) 10, 13
2. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a "Siamese" time delay neural network. In: NIPS (1993) 6
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT press (2009) 4
4. Diaconis, P., Graham, R.L.: Spearman's footrule as a measure of disarray. Journal of the Royal Statistical Society: Series B (Methodological) **39**(2), 262–268 (1977) 10
5. Diaz, R., Marathe, A.: Soft labels for ordinal regression. In: CVPR (2019) 1, 10, 12, 13
6. Jech, T.J.: The Axiom of Choice. Courier Corporation (2008) 4
7. Kahn, A.B.: Topological sorting of large networks. Communications of the ACM **5**(11), 558–562 (1962) 2, 4, 7, 11, 13, 14
8. Knuth, D.E., Szwarcfiter, J.L.: A structured program to generate all topological sorting arrangements. Information Processing Letters **2**(6), 153–157 (1974) 2, 4
9. Lee, S.H., Kim, C.S.: Deep repulsive clustering of ordered data based on order-identitiy decomposition. In: ICLR (2021) 1, 3, 5, 9
10. Levi, G., Hassner, T.: Age and gender classification using convolutional neural networks. In: CVPR Workshops (2015) 3, 10
11. Li, W., Huang, X., Lu, J., Feng, J., Zhou, J.: Learning probabilistic ordinal embeddings for uncertainty-aware regression. In: CVPR (2021) 1, 10, 12, 13
12. Lim, K., Shin, N.H., Lee, Y.Y., Kim, C.S.: Order learning and its application to age estimation. In: ICLR (2020) 1, 3, 5, 9, 10, 11, 12, 13, 14
13. Liu, X., Li, S., Ge, Y., Ye, P., You, J., Lu, J.: Recursively conditional gaussian for ordinal unsupervised domain adaptation. In: ICCV (2021) 2, 9
14. Liu, Y., Kong, A.W.K., Goh, C.K.: A constrained deep neural network for ordinal regression. In: CVPR (2018) 12
15. Liu, Y., Wang, F., Kong, A.W.K.: Probabilistic deep ordinal regression based on Gaussian processes. In: CVPR (2019) 12
16. Niu, Z., Zhou, M., Wang, L., Gao, X., Hua, G.: Ordinal regression with multiple output CNN for age estimation. In: CVPR (2016) 12
17. Pan, H., Han, H., Shan, S., Chen, X.: Mean-variance loss for deep age estimation from a face. In: CVPR (2018) 1
18. Parikh, D., Grauman, K.: Relative attributes. In: ICCV (2011) 3
19. Pearce, D.J., Kelly, P.H.: A dynamic topological sort algorithm for directed acyclic graphs. Journal of Experimental Algorithmics **11**, 1–7 (2007) 2, 4
20. Reingold, E.M., Nievergelt, J., Deo, N.: Combinatorial Algorithms: Theory and Practice. Prentice Hall (1977) 2, 4
21. Ricanek, K., Tesafaye, T.: MORPH: A longitudinal image database of normal adult age-progression. In: FGR (2006) 10
22. Schifanella, R., Redi, M., Aiello, L.M.: An image is worth more than a thousand favorites: Surfacing the hidden beauty of Flickr pictures. In: ICWSM (2015) 10
23. Schröder, B.S.W.: Ordered Sets: An Introduction. Springer (2003) 4
24. Shin, N.H., Lee, S.H., Kim, C.S.: Moving window regression: A novel approach to ordinal regression. In: CVPR (2022) 1, 3, 5

25. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: FixMatch: Simplifying semi-supervised learning with consistency and confidence. In: NIPS (2020) 2, 9, 12
26. Souri, Y., Noury, E., Adeli, E.: Deep relative attributes. In: ACCV (2016) 3, 11, 14
27. Tarjan, R.: Finding dominators in directed graphs. SIAM Journal on Computing **3**(1), 62–89 (1974) 4
28. Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., Shinozaki, T.: FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling. In: NIPS (2021) 2, 9, 12
29. Zhang, Y., Liu, L., Li, C., Loy, C.C.: Quantifying facial age by posterior of age comparisons. In: BMVC (2017) 1, 5
30. Zou, Y., Yang, X., Yu, Z., Kumar, B., Kautz, J.: Joint disentangling and adaptation for cross-domain person re-identification. In: ECCV (2020) 2, 9