APSIPA Transactions on Signal and Information Processing, 2023, 12, e34

Overview Paper

Order Learning – An Overview

Seon-Ho Lee, Nyeong-Ho Shin and Chang-Su Kim*

School of Electrical Engineering, Korea University, Seoul, Korea

ABSTRACT

Order learning aims to learn the ordering relationship among objects by comparing them. Recently, several order learning techniques have achieved great performances on various computer vision tasks. In this paper, we provide an overview of these order learning techniques. First, we briefly discuss conventional rank estimation algorithms related to order learning. Second, we review the order learning techniques in detail. Third, we discuss the results of order learning on three vision applications: facial age estimation, historical color image (HCI) classification, and aesthetic quality assessment.

Keywords: Order learning, ordinal regression, rank estimation.

1 Introduction

In rank estimation, we attempt to find the rank (or ordered class) of an object instance. It is different from ordinary classification, for its classes are arranged in a natural order. For example, in online user ratings of a product or service, classes can be ordered from 'excellent' to 'good,' 'satisfactory,' 'poor,' and 'very poor.' Rank estimation is widely used for various computer vision tasks, including facial age estimation [35], aesthetic quality assessment [37], and HCI classification [31].

With the great success of convolutional neural networks (CNNs) [42], many CNN-based techniques [17, 22, 26, 35, 43, 44] have been proposed for

*Corresponding author: Chang-Su Kim, changsukim@korea.ac.kr.

Received 25 April 2023; Revised 10 July 2023
ISSN 2048-7703; DOI 10.1561/116.00000226

rank estimation. Most such techniques [22, 26] adopt the ordinal regression framework in which the rank of an instance is directly estimated via a classifier or a regressor. On the other hand, some algorithms [35, 44] regard rank estimation as an ordinary classification problem. Also, Lee and Kim [17] and Souri *et al.* [43] exploit the relationship between instances via pairwise comparison. However, rank estimation is still challenging because there is no clear distinction between ranks in many cases. For example, in facial age estimation, the aging process has large individual differences, so some people look younger or older than their actual ages.

Recently, several order learning algorithms, which exploit order relations between instances, have been proposed for rank estimation, providing excellent results. In [24], Lim *et al.* first proposed the notion of order learning, which learns ordering relationship between objects and determines the rank of an unseen object by comparing it with multiple references with known ranks. It provides promising results since relative assessment is easier than absolute assessment in general. In [18], Lee *et al.* improved the performance of order learning by finding optimal references for comparison. Also, Shin *et al.* [41] introduced a regression framework for order learning, and Lee *et al.* [20] simplified the training and testing processes of order learning by constructing a well-sorted embedding space. Moreover, Lee and Kim [19] developed a weakly supervised order learning scheme by extending topological sorting.

To the best of our knowledge, there is no review paper to provide a holistic view of order learning. Thus, in this paper, we review these order learning algorithms in detail. In Section 2, we first summarize conventional rank estimation approaches related to order learning. Then, we explain the order learning algorithms in Section 4. Section 5 describes the results of order learning on various applications, including age estimation, HCI classification, and aesthetic quality assessment. Lastly, Section 7 concludes this overview paper.

2 Related Work

Various techniques have been developed to estimate the ordinal class or rank of an object. As in Figure 1, they can be grouped into three categories: classification, ordinal regression, and pairwise comparison.

2.1 Classification

Classification algorithms [35, 44] predict ranks by training a classifier to categorize each instance into one of discrete ordinal classes, as in ordinary classification [42]. Figure 2(a) illustrates the framework of classification algorithms. In [35], tens of classifiers are trained and their predictions are averaged

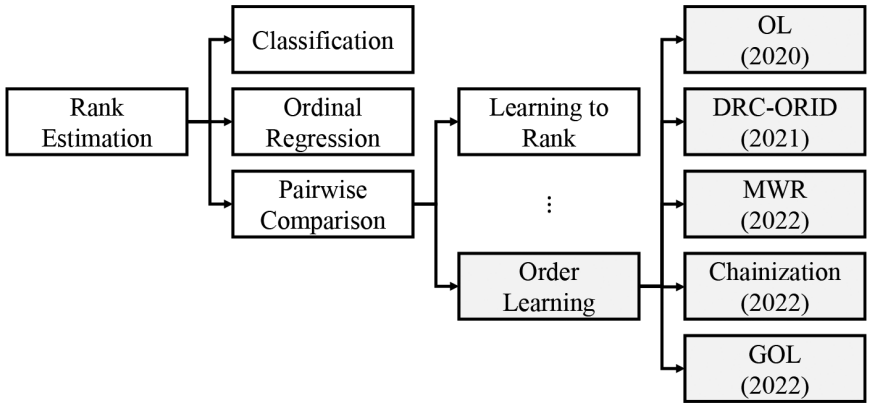
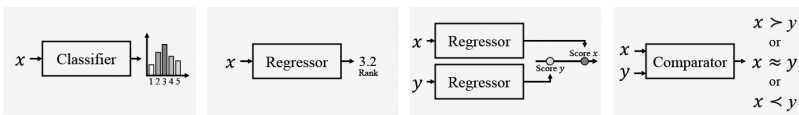


Figure 1: Categorization of rank estimation algorithms.

to yield the output. In [44], the entire rank range is covered by overlapping rank intervals so that each rank belongs to several intervals. Then, binary classifiers are trained to predict whether an instance belongs to each interval or not. However, typical losses for classification, such as cross-entropy, may not be optimal for rank estimation. Unlike ordinary classification, different errors have different levels of severity in rank estimation. But, regardless of whether the rank of 13 is mistaken as 6 or 70, they may be considered as the same amount of errors in typical classification losses. Therefore, by its design, the classification approach may be sub-optimal for rank estimation.

2.2 Ordinal Regression

Ordinal regression methods estimate the rank of an instance directly using classifiers or regressors. Figure 2(b) shows the overall framework of ordinal regression. In [9, 30], multiple binary classifiers are used, and each classifier predicts whether the rank of an instance is higher than a specific rank or not. Then, the rank is estimated by combining the binary classification results. In [27], a simple ordinal regressor is trained for small datasets. In [47], a regressor is employed to estimate the rank from multi-scale patches of an input instance.



(a) (b) (c) (d)

Figure 2: Different frameworks for rank estimation: (a) classification, (b) ordinal regression, (c) LTR, and (d) order learning.

However, in many cases, the direct estimation of ranks is challenging even for human beings. Hence, in [10], soft labels are used to train a regressor more reliably by avoiding too big penalties on close estimates. Note that there are several review papers [13, 45] on ordinal regression. On the other hand, we review the order learning algorithms for the first time in this paper.

2.3 Pairwise Comparison

Pairwise comparison also has been used to estimate object ranks because relative evaluation is easier than absolute evaluation in general. In aesthetic quality assessment, Lee and Kim [17] predict the aesthetic score ratio of a given instance pair and apply the Saaty's scaling method [36] to regress an absolute score. In age estimation, some learning-to-rank (LTR) methods [5, 6, 43] use a ranking network, which outputs the relative priorities of instances, as shown in Figure 2(c).

Order learning belongs to this pairwise comparison category and is similar to LTR in that both approaches attempt to estimate ordering relationship between instances. However, whereas LTR compares instances during the network training only, order learning estimates the rank of a test instance by comparing it with reference instances with known ranks. Figure 2(d) illustrates the framework of order learning.

3 Preliminary – Order

Mathematically, *order* [38] is a binary relation, often denoted by $\leq$, on a set $\Theta = \{\theta_0, \theta_1, \ldots, \theta_{M-1}\}$, which 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 $\theta_i \leq \theta_k$.

In practice, an order describes the priorities of classes (or ranks), where each class θ_i represents one or more object instances. For example, in age estimation, θ_i may represent i-year-olds, and $\theta_{17} < \theta_{35}$ represents that 17-year-olds are younger than 35-year-olds. Let $\theta(\cdot)$ be the class function, and let x and y be instances. Then, for example, $\theta(x) = \theta_{20}$ means that person x is 20-year-old. Also, for two object instances x and y, their ordering relationship is defined according to their class difference as

$$\begin{aligned} x \succ y \text{ if } \theta(x) - \theta(y) > \tau, \\ x \approx y \text{ if } |\theta(x) - \theta(y)| \leq \tau, \\ x \prec y \text{ if } \theta(x) - \theta(y) < -\tau, \end{aligned} \tag{1}$$

where τ is a threshold. We use the expression *ordering* to describe instance relations, while using *order* exclusively for class relations. Also, 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.

4 Order Learning Algorithms

4.1 Order Learning

4.1.1 Pairwise Comparison

OL [24] is the first algorithm for order learning. It is based on the idea that it is easier to predict ordering relationship between instances than to estimate their ranks directly; telling the older one between two people is easier than estimating their exact ages. Therefore, order learning attempts to learn the pairwise ordering relationship from training data. To this end, Lim *et al.* [24] developed a pairwise comparator, which predicts the ordering relationship between two input instances x and y. As shown in Figure 3, the comparator consists of an encoder and a ternary classifier. The encoder $h_x = h(x)$, which is a VGG16 network without fully connected layers, maps an input instance x to a feature vector h_x. Also, it extracts the feature vector $h_y = h(y)$ from the other instance y. Then, the classifier, which is implemented by a series of fully connected layers, takes h_x and h_y as its input and yields the softmax probabilities for the ordering relationship $p^{xy} = [p^{xy}_{\prec}, p^{xy}_{\approx}, p^{xy}_{\succ}]^t$. Given a training instance pair (x, y), the comparator is trained to minimize the cross-entropy loss

$$l_{\text{ce}}(x, y) = \sum q^{xy}_{\prec} \log p^{xy}_{\prec} + q^{xy}_{\approx} \log p^{xy}_{\approx} + q^{xy}_{\succ} \log p^{xy}_{\succ} \tag{2}$$

where $q^{xy} = [q^{xy}_{\prec}, q^{xy}_{\approx}, q^{xy}_{\succ}]^t$ is the ground-truth one hot vector.

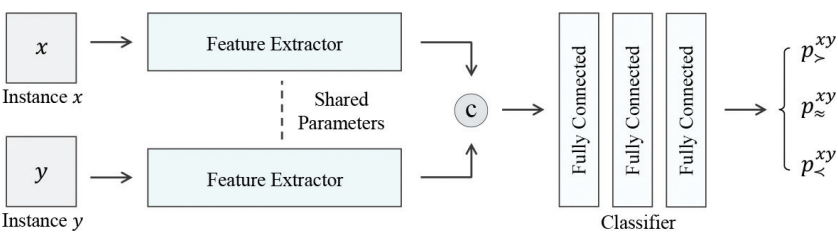


Figure 3: An overview of the pairwise comparator, where ⓒ denotes concatenation.

4.1.2 Rank Estimation

OL estimates the class $\theta(x)$ of a test instance x by comparing it with m reference instances $y_0, y_1, \ldots, y_{m-1}$. The references are selected from the training set $\mathcal{X}$ based on the reliability scores, given by

$$s(y_i) = \sum_{y_j \in \mathcal{X}, j \neq i} -l_{\text{ce}}(y_i, y_j). \tag{3}$$

For each class θ_i, OL selects the $m/|\Theta|$ instances with the highest reliability scores as the references. Note that the ground-truth ranks of the reference instances are known, for they are selected from the training set.

After selecting the references, the comparator predicts the ordering relationship between x and y_i. If θ' is an estimate of the true class $\theta(x)$, the consistency between the comparator result and the estimate is defined as

$$\begin{aligned}\phi_{\text{con}}(x, y_i, \theta') = [x \prec y_i][\theta' - \theta(y_i) > \tau] + [x \approx y_i][|\theta' - \theta(y_i)| \leq \tau] \\ + [x \succ y_i][\theta' - \theta(y_i) < -\tau]\end{aligned} \tag{4}$$

where $[\cdot]$ is the indicator function. The consistency function ϕ_{con} outputs either 0 for an inconsistent case or 1 for a consistent case. Note that an inconsistent case can occur due to the error in the comparator prediction and/or the error in the estimate. OL determines $\hat{\theta}(x) = \theta'$ such that θ' maximizes the consistency with all references, given by

$$\hat{\theta}(x) = \underset{\theta' \in \Theta}{\operatorname{argmax}} \sum_{i=0}^{m-1} \phi_{\text{con}}(x, y_i, \theta'). \tag{5}$$

This is called the maximum consistency (MC) rule.

4.1.3 K Chain Hypothesis

In the default mode of OL, it is assumed that ordered classes form a single chain (1CH). However, in the K chain hypothesis (KCH), the classes are assumed to form K disjoint chains. Note that a chain denotes a linearly ordered and maximal subset of an ordered set. Therefore, classes in different chains are incomparable. For example, in age estimation, chains can be divided according to the gender in {female, male} and the ethnic group in {African, Asian, European}. This is because it is difficult to compare the ages of people of different genders or in different ethnic groups. Thus, people in different chains are assumed incomparable for age estimation. In this case, there are 6 chains in total. Also, 2CH or 3CH can be obtained by considering genders or ethnic groups only.

In KCH, the comparator is trained similarly to 1CH. However, only the pairs of instances belonging to the same chain are used for training. Also, a K-way chain classifier is trained additionally, which predicts the chain that an input instance belongs to. During the test, given an input instance, OL determines its chain using the chain classifier, compares it with the references in the same chain, and then estimates its class using the MC rule.

4.2 DRC-ORID

4.2.1 Motivation

In OL [24], additional information, such as gender or race in age estimation, is needed to divide an ordered dataset into multiple chains reliably. However, there are many datasets in which such information is not available. Therefore, Lee and Kim [18] proposed a new chain-division scheme called DRC-ORID. It is based on the idea that some characteristics of objects are not related to their ranks, and the ranks of objects sharing such characteristics can be compared more reliably; for example, it is easier to tell the older one between people of the same gender than between people of different genders.

4.2.2 Order-Identity Decomposition

DRC-ORID [18] aims to partition an ordered dataset $\mathcal{X}$ into k clusters by grouping the elements according to their characteristics unrelated to their ranks. These characteristics are referred to as 'identity' features. In general, object instances can be compared more easily when they have more similar identity features irrelevant to order. Therefore, DRC-ORID decomposes the information of each instance into an order feature and an identity feature exclusively. To this end, it uses the order-identity decomposition (ORID) network, which is composed of three parts: autoencoder, classifier, and discriminator. Figure 4 shows the architecture of the ORID network.

First, in autoencoder, the encoder, $h_x = f(x)$, extracts feature vector h_x from input instance x, while the decoder, $\hat{x} = g(h_x)$, reconstructs $\hat{x}$ from h_x. The autoencoder is trained by minimizing the reconstruction loss $||x - \hat{x}||$.

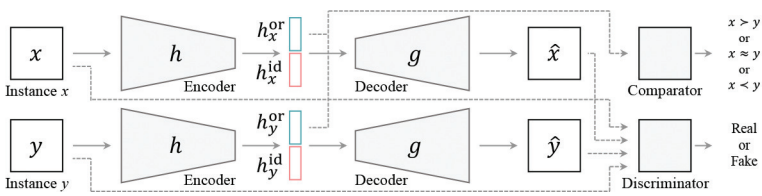


Figure 4: An overview of the ORID network. This figure is excerpted from [18].

Also, the first d_{or}-dimensional sub-vector in $h_x \in \mathbb{R}^{d_{\text{or}}+d_{\text{id}}}$ is regarded as the order feature h_x^{or}. The last d_{id}-dimensional sub-vector in h_x is normalized and regarded as the identity feature h_x^{id}.

To divide order-related information and its complementary information into h_x^{or} and h_x^{id} respectively, a classifier is employed. For a pair of instances x and y, it classifies their ordering relationship into one of three categories in (1) from the order features h_x^{or} and h_y^{or}. It is trained using the cross-entropy loss, as done in [24]. Note that the autoencoder and classifier are trained jointly. Hence, to minimize both the reconstruction loss and the cross-entropy loss with the limited capacity of h_x, the information deciding the ordering relationship tends to be encoded into the order features. On the other hand, the remaining information necessary for the reconstruction of $\hat{x}$ and $\hat{y}$ are encoded into the identity features h_{id}^x and h_{id}^y. Also, the discriminator that tells real images from reconstructed images helps the decoder to reconstruct a more realistic output $\hat{x}$.

4.2.3 Deep Repulsive Clustering

After obtaining the identity features of all instances $x \in \mathcal{X}$, they partition them into k clusters $C_0, C_1, \ldots, C_{k-1}$. They measure the overall quality of clustering by

$$J(\{\mathcal{C}_j\}_{j=0}^{k-1}, \{c_j\}_{j=0}^{k-1}) = \sum_{j=0}^{k-1} \sum_{x \in \mathcal{C}_j} \left((h_x^{\text{id}})^t c_j - \alpha \frac{1}{k-1} \sum_{l \neq j} (h_x^{\text{id}})^t c_l \right) \tag{6}$$

where the first term is the similarity of an instance in $\mathcal{C}_j$ to the centroid c_j, the second term with the negative sign quantifies the average dissimilarity of the instance from the other centroids, and α is a nonnegative weight. For high-quality clustering, each instance should be located around the centroid of its cluster and be far from the other clusters.

To find the clustering result maximizing the objective function J, they use an iterative algorithm called DRC,

1. Centroid rule: After fixing the clusters $\{\mathcal{C}_j\}_{j=0}^{k-1}$, DRC updates the centroids $\{c_j\}_{j=0}^{k-1}$ to maximize J in (6). Since identity features are normalized and lie on the unit sphere, DRC constrains all centroids also to be on the unit sphere. Then, the optimization problem is given by

$$\text{maximize } J(\{c_j\}_{j=1}^{k}) \text{ subject to } c_j^t c_j = 1 \text{ for all } j = 0, \ldots, k-1. \tag{7}$$

Using Lagrangian multipliers [2], the optimal centroids are obtained as

$$c_j = \frac{\left(\sum_{x \in \mathcal{C}_j} h_x^{\text{id}} - \alpha \frac{1}{k-1} \sum_{x \in \mathcal{X} \setminus \mathcal{C}_j} h_x^{\text{id}} \right)}{\left\| \sum_{x \in \mathcal{C}_j} h_x^{\text{id}} - \alpha \frac{1}{k-1} \sum_{x \in \mathcal{X} \setminus \mathcal{C}_j} h_x^{\text{id}} \right\|}. \tag{8}$$

2. Nearest neighbor (NN) rule: On the other hand, after fixing the centroids, DRC updates the membership of each instance to maximize J in (6). The optimal cluster $\mathcal{C}_j$ is given by

$$\mathcal{C}_j = \left\{x \,|\, (h_x^{\text{id}})^t c_j \geq (h_x^{\text{id}})^t c_l \text{ for all } 0 \leq l \leq k-1\right\}. \tag{9}$$

   In other words, an instance should be assigned to $\mathcal{C}_j$ if its nearest centroid is c_j.

DRC applies the centroid rule and the NN rule iteratively until convergence. Also, the traning and clustering of the ORID network are alternately repeated.

4.2.4 Rank Estimation

To estimate the rank of an unseen test instance x, DRC-ORID first extracts the identity feature h_x^{id} using the ORID encoder. Then, by comparing h_x^{id} with the centroids $\{c_j\}_{j=0}^{k-1}$ based on the NN rule, it finds the most similar centroid c_l. Then, x is declared to belong to cluster $\mathcal{C}_l$. Without loss of generality, let us assume that the classes (or ranks) are the first m natural numbers, $\Theta = \{1, 2, \dots m\}$. For each $i \in \Theta$, DRC-ORID selects a reference instance y_i with rank i from cluster $\mathcal{C}_l$, so that it is the most similar to x. Specifically,

$$y_i = \underset{y \in \mathcal{C}_l \,:\, \theta(y)=i}{\operatorname{argmax}} (h_x^{\text{id}})^t h_y^{\text{id}}. \tag{10}$$

Then, by comparing x with y_i, the comparator yields the probability vector $p^{xy_i} = (p_{\succ}^{xy_i}, p_{\approx}^{xy_i}, p_{\prec}^{xy_i})$ for the three cases in (1). Thus, given y_i, the probability of $\theta(x) = \theta'$ can be written as

$$\begin{aligned} P_{\theta(x)}(\theta' \,|\, y_i) = p_{\succ}^{xy_i} \cdot P_{\theta(x)}(\theta' \,|\, x \succ y_i) + p_{\approx}^{xy_i} \cdot P_{\theta(x)}(\theta' \,|\, x \approx y_i) \\ + p_{\prec}^{xy_i} \cdot P_{\theta(x)}(\theta' \,|\, x \prec y_i). \end{aligned} \tag{11}$$

If $x \succ y_i$, $\theta(x) - \theta(y_i) = \theta' - i > \tau$ from (1). Also, the maximum possible rank is m. Hence, it is assumed that $\theta(x)$ has the uniform distribution between $i + \tau + 1$ and m. In other words,

$$P_{\theta(x)}(\theta' \,|\, x \succ y_i) \sim U(i + \tau + 1, m) \tag{12}$$

where U denotes a discrete uniform distribution. Similarly, it is assumed that $P_{\theta(x)}(\theta' \,|\, x \approx y_i) \sim U(i-\tau, i+\tau)$ and $P_{\theta(x)}(\theta' \,|\, x \prec y_i) \sim U(1, i-\tau-1)$. Then, DRC-ORID has the *a posteriori* probability $P_{\theta(x)}(\theta' \,|\, y_1, \dots y_m)$ by averaging those single-reference inferences in (11);

$$P_{\theta(x)}(\theta' \,|\, y_1, \dots y_m) = \frac{1}{m} \sum_{i=1}^{m} P_{\theta(x)}(\theta' \,|\, y_i). \tag{13}$$

Finally, the MAP estimate of the rank of x is obtained by

$$\hat{\theta}(x) = \underset{\theta' \in \Theta^l}{\operatorname{argmax}} P_{\theta(x)}(\theta' \,|\, y_1, \dots y_m). \tag{14}$$

4.3 Moving Window Regression

4.3.1 Motivation

Unlike OL, which is based on ternary classification, the moving window regression (MWR) algorithm [41] exploits a continuous regression score, which is obtained by comparing the test instance with two references at once. Hence, MWR can yield a more precise order relationship than OL. In this section, we provide details about the MWR algorithm.

4.3.2 ρ-Rank

MWR uses a continuous regression score, called ρ-rank

$$\rho(x, y_1, y_2) = \frac{\theta(x) - \mu(y_1, y_2)}{\tau(y_1, y_2)} \tag{15}$$

where y_1 and y_2 are two references with $\theta(y_1) \leq \theta(x) \leq \theta(y_2)$. Also, $\mu(y_1, y_2) = \frac{1}{2}(\theta(y_1) + \theta(y_2))$ is the average rank of the two references, and $\tau(y_1, y_2) = \frac{1}{2}(\theta(y_2) - \theta(y_1))$ is half of the rank difference between them. The ρ-rank quantifies the ordinal relations among the ranks of input and reference instances: it measures how much greater the input is than the first reference and how much smaller it is than the second reference. Note that $\rho \in [-1, 1]$. From the the ρ-rank, the absolute rank θ can be reconstructed by

$$\theta(x) = \rho(x, y_1, y_2) \cdot \tau(y_1, y_2) + \mu(y_1, y_2). \tag{16}$$

Hence, after predicting the ρ-rank, θ is obtained via (16).

4.3.3 ρ-Regressor

To estimate the ρ-rank in (15), ρ-regressor, which consists of an encoder $h(\cdot)$ and a regression module $g(\cdot)$, is employed. Since the ρ-rank of x is determined in the context of two references y_1 and y_2, the encoder extracts the features $h(y_1)$ and $h(y_2)$ as well as the feature $h(x)$. Then, the regression module takes the triplet $(h(x), h(y_1), h(y_2))$ and obtains an estimate of the ρ-rank in (15), which is given by

$$\hat{\rho}(x, y_1, y_2) = g(h(x), h(y_1), h(y_2)). \tag{17}$$

To train the ρ-regressor effectively and lower the learning difficulty, a triplet (x, y_1, y_2) is formed by selecting the references with a fixed

$$\tau = \frac{1}{2}(\theta(y_2) - \theta(y_1)). \tag{18}$$

By fixing τ, the ρ-regressor needs to consider a much smaller subset of $\{(x, y_1, y_2)\}$ and can achieve more reliable regression. The ρ-regressor is trained to minimize the squared error between the ground-truth ρ and the estimate $\hat{\rho}$. When $\theta(x) < \theta(y_1)$ or $\theta(x) > \theta(y_2)$, the ground-truth ρ is set to -1 or 1, respectively.

4.3.4 Moving Window Regression

To predict the absolute rank $\hat{\theta}(x)$ of an unseen test instance x, the process that moves a window $[\theta(y_1), \theta(y_2)]$ and estimates the ρ-rank $\hat{\rho}(x, y_1, y_2)$ in (17) using the ρ-regressor is performed iteratively. This process is called MWR.

First, an initial prediction is performed to obtain an initial estimate $\hat{\theta}^0(x)$, where the superscript indicates the iteration index. In Figure 5(a), the initial prediction process is visualized. After the feature vector $f(x)$ is extracted from the encoder, the K NNs are found among all training instances in terms of the Euclidean distances in the feature space. Then, the initial estimate $\hat{\theta}^0(x)$ is yielded by averaging the ranks of these neighbors. Note that feature vectors are sorted according to their ranks in the feature space, as visualized in Figure 5(b). Therefore, this prediction procedure is reasonable.

Next, the estimate is iteratively refined, as in Figure 5(c). At iteration step t, the previous estimate $\hat{\theta}^{t-1}(x)$ is refined to $\hat{\theta}^t(x)$. A pair of references y_1^t and y_2^t, whose ranks are $\theta(y_1^t) = \hat{\theta}^{t-1}(x) - \tau$ and $\theta(y_2^t) = \hat{\theta}^{t-1}(x) + \tau$, are chosen among the training instances. Note that the search window is designated as $[\theta(y_1^t), \theta(y_2^t)]$ and centered around the previous estimate $\hat{\theta}^{t-1}(x)$. Within the

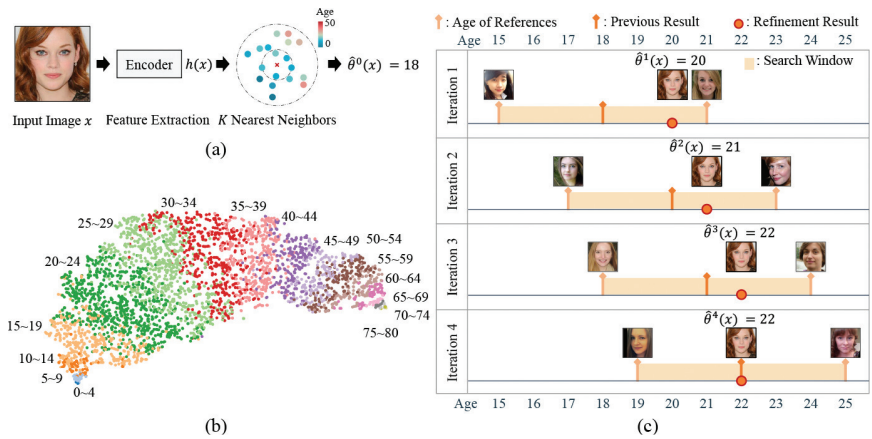


Figure 5: An example of the MWR process in facial age estimation when the ground-truth age of input x is 22 and τ equals 3: (a) initial prediction, (b) t-SNE visualization of the embedding space, and (c) iterative MWR refinement. This figure is excerpted from [41].

search window, the ρ-rank $\hat{\rho}(x, y_1^t, y_2^t)$ is yielded by the ρ-regressor. Then, $\hat{\rho}(x, y_1^t, y_2^t)$ is converted to

$$\begin{aligned}\hat{\theta}^t(x) &= \text{round}\big(\hat{\rho}(x, y_1^t, y_2^t) \cdot \tau(y_1^t, y_2^t) + \mu(y_1^t, y_2^t)\big) \\ &= \text{round}\big(\hat{\rho}(x, y_1^t, y_2^t) \cdot \tau + \hat{\theta}^{t-1}(x)\big). \end{aligned} \tag{19}$$

The equality in (19) holds since τ is a fixed value and $\mu(y_1^t, y_2^t)\big) = \hat{\theta}^{t-1}(x)$. The iterative MWR process is terminated when $\hat{\theta}^t(x) = \hat{\theta}^{t-1}(x)$ is satisfied, as shown in Iteration 4 in Figure 5(c), or a predefined number of iterations is reached.

4.3.5 Reference Selection

To estimate the rank of a test instance effectively, it is compared with selected reference pairs. Those are chosen from the training set before testing, based on the regression error γ_{e}, which corresponds to the average estimation error of ρ-ranks, when (y_1, y_2) is employed as the reference pair. The regression error γ_{e} is defined as

$$\gamma_{\text{e}}(y_1, y_2) = \frac{1}{|W|} \sum_{x \in W} |\hat{\rho}(x, y_1, y_2) - \rho(x, y_1, y_2)| \tag{20}$$

where $W = \{x \,|\, \theta(x) \in [\theta(y_1) - \alpha, \theta(y_2) + \alpha]\}$. Hence, at iteration t during the MWR process, for reliable regression, the optimal reference pair (y_1, y_2) with the lowest $\gamma_{\text{e}}(y_1, y_2)$, which satisfies the constraints of $\theta(y_1) = \hat{\theta}^{t-1}(x) - \tau$ and $\theta(y_2) = \hat{\theta}^{t-1}(x) + \tau$, is selected.

4.4 Geometric Order Learning

4.4.1 Motivation

In rank estimation, an order and a metric convey complementary information: the order provides directional information between ranks, while the metric does length (or magnitude of difference) information. However, previous order learning algorithms, including OL and DRC-ORID, only consider the ordering relationship between two instances x and y and disregard how much x is different from y. In other words, it ignores metric information. In contrast, geometric order learning (GOL) [20] exploits both order and metric relations to estimate the rank of an instance more reliably. Figure 6 is an overview of GOL.

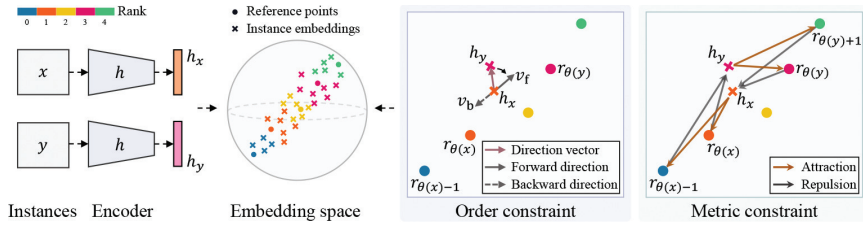


Figure 6: An overview of GOL algorithm. This figure is from [20].

4.4.2 Embedding Space Construction

GOL aims to construct an embedding space in which the direction and distance between instances represent their order and metric relations. To this end, GOL adopts two types of geometric constraints in an embedding space: the order constraint and the metric constraint. Specifically, the order constraint sorts instances directionally according to the ranks, while the metric constraint separates two instances further if their rank difference is larger.

Order constraint: Suppose that there are m ranks in a training set $\mathcal{X}$. Without loss of generality, the ranks are assumed to be consecutive integers in $\Theta = \{0, 1, \ldots, m-1\}$. As in other order learning techniques, GOL maps each instance $x \in \mathcal{X}$ into a feature vector $h_x = h(x)$ in an embedding space. Also, the output of the last pooling layer in the encoder h is normalized so that $h_x^t h_x = 1$.

The order constraint encourages instances to be sorted according to their ordering relationships. In other words, for two instances x and y with ordering $x \prec y$, the vector from h_x to h_y should be aligned with the direction of the rank increment in the embedding space. To model such rank directions, GOL introduces m reference points, $r_0, r_1, \ldots, r_{m-1}$, which are learnable parameters and guide the positions of the m ranks in the embedding space.

The direction vector $v(r, s) = (s - r)/\|s - r\|$ represents the direction from point r to point s on the unit hypersphere. Then, $v(r_i, r_j)$ is called the rank direction from rank i to rank j. Also, the rank direction $v(r_i, r_j)$ is forward if $i < j$, and backward if $i > j$. Note that forward directions may differ from one another.

If $x \prec y$, GOL determines the forward and backward rank directions as

$$v_{\mathrm{f}} = v(r_{\theta(x)}, r_{\theta(y)}), \tag{21}$$

$$v_{\mathrm{b}} = v(r_{\theta(x)}, r_{\theta(x)-1}). \tag{22}$$

Then, the encoder is trained so that the embedded features h_x and h_y satisfy the order constraint:

$$x \prec y \quad \Leftrightarrow \quad v_{\mathrm{f}}^t v(h_x, h_y) > v_{\mathrm{b}}^t v(h_x, h_y). \tag{23}$$

In other words, the direction vector $v(h_x, h_y)$ should be aligned more with the forward direction v_f than with the backward direction v_b.

To enforce the order constraint in (23), we compute the softmax probability $p^{xy} = [p_\mathrm{f}^{xy}, p_\mathrm{b}^{xy}]^t$, where

$$p_\mathrm{f}^{xy} = \frac{e^{v_\mathrm{f}^t v(h_x,h_y)}}{e^{v_\mathrm{f}^t v(h_x,h_y)} + e^{v_\mathrm{b}^t v(h_x,h_y)}} \tag{24}$$

and $p_\mathrm{b}^{xy} = 1 - p_\mathrm{f}^{xy}$. We then define the order loss L_order as the cross entropy between p^{xy} and $q^{xy} = [q_\mathrm{f}^{xy}, q_\mathrm{b}^{xy}]^t = [1, 0]^t$, given by

$$L_\mathrm{order} = q_\mathrm{f}^{xy} \log p_\mathrm{f}^{xy} + q_\mathrm{b}^{xy} \log p_\mathrm{b}^{xy}. \tag{25}$$

The order loss for case $x \succ y$ is formulated similarly in a symmetric manner.

Metric constraint: A metric constraint aims to make the distance between instances in the embedding space reflect their rank difference. Specifically, in GOL, the metric constraint is defined as

$$|\theta(x) - \theta(y)| > \tau \quad \Leftrightarrow \quad d_\mathrm{e}(h_x, h_y) > \delta \tag{26}$$

where d_e is the Euclidean distance in the embedding space, and δ is a margin. Since $|\theta(x) - \theta(y)| > \tau$ means that either $x \prec y$ or $x \succ y$, the metric constraint in (26) is equivalent to

$$x \approx y \quad \Leftrightarrow \quad d_\mathrm{e}(h_x, h_y) \leq \delta. \tag{27}$$

If $x \prec y$, to satisfy the metric constraint in (26), GOL defines a loss $L_{x \prec y}$ as

$$\begin{aligned} L_{x \prec y} = & \sum_{i : i \leq \theta(x)} \max(d_\mathrm{e}(r_i, h_x) - d_\mathrm{e}(r_i, h_y) + \delta, 0) \\ & + \sum_{j : j \geq \theta(y)} \max(d_\mathrm{e}(r_j, h_y) - d_\mathrm{e}(r_j, h_x) + \delta, 0). \end{aligned} \tag{28}$$

To minimize the first sum, $d_\mathrm{e}(r_i, h_x)$ should be reduced, while $d_\mathrm{e}(r_i, h_y)$ should be increased. Thus, reference points $r_i, 0 \leq i \leq \theta(x)$, are trained to attract h_x and repel h_y. The second sum in (28) is similar. $L_{x \succ y}$ is formulated symmetrically.

On the other hand, to encourage the constraint in (27), GOL uses another loss

$$L_{x \approx y} = \sum_{i \in \Theta} \max(|d_\mathrm{e}(r_i, h_x) - d_\mathrm{e}(r_i, h_y)| - \delta, 0). \tag{29}$$

Readers more interested in the derivation of the metric loss are referred to [20].

Loss function: In addition to the order loss and the metric loss, GOL employs the center loss,

$$L_{\text{center}} = d_{\text{e}}(r_{\theta(x)}, h_x) + d_{\text{e}}(r_{\theta(y)}, h_y). \tag{30}$$

Finally, the encoder parameters and the reference points r_i are jointly optimized by minimizing the total loss, given by

$$L_{\text{total}} = L_{\text{order}} + L_{\text{metric}} + L_{\text{center}}. \tag{31}$$

Notice that reference points play essential roles in embedding space construction. Specifically, in L_{order}, they guide forward and backward rank directions, so that instances are sorted directionally according to the ranks. In L_{metric}, reference points attract and repel instances to satisfy the metric constraint. In other words, GOL uses the reference points to satisfy both order and metric constraints simultaneously and thus construct a well-arranged, well-clustered embedding space.

4.4.3 Rank Estimation

For rank estimation, GOL uses the simple k-NN rule for rank estimation. Hence, unlike other order learning algorithms, it does not need a complex reference selection process. Given a test instance x, in the embedding space, GOL finds a set $\mathcal{N}$ of its k NNs among all training instances in $\mathcal{X}$. Then, the rank of x is estimated by

$$\hat{\theta}(x) = \frac{1}{k} \sum_{y \in \mathcal{N}} \theta(y). \tag{32}$$

4.5 Chainization

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. Here, $\mathcal{X}$ is a set of n training instances. Other order learning algorithms [18, 20, 24, 41] assume that the ordering relation for every pair of training instances is known. In other words, it is assumed that $\mathcal{L}$ is a *linear ordering* of instances:

$$(x, y) \in \mathcal{L} \text{ or } (y, x) \in \mathcal{L} \text{ for all } x, y \in \mathcal{X}. \tag{33}$$

Note that both (x, y) and (y, x) belong to $\mathcal{L}$ if $x \approx y$.

However, such complete ordering information is not always available. Chainization [19] assumes that only the binary ordering information between some selected pairs of instances may be available. In other words, a *partial ordering* of instances is given,

$$\mathcal{P} = \{(x, y) : \text{It is known that } x \preccurlyeq y\} \subset \mathcal{L}. \tag{34}$$

The chainization algorithm aims to linearly extend or 'chainize' a partial ordering $\mathcal{P}$ to a linear ordering $\mathcal{L}$. Furthermore, if $\mathcal{L}$ is estimated reliably, order learning performance can be enhanced by using $\mathcal{L}$ as auxiliary information for training. Let us describe the chainization process in detail.

Graph representation of ordering: A partial ordering $\mathcal{P}$ of instances in $\mathcal{X}$ is represented by a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and $\mathcal{E} = \{(v_i, v_j) : (x_i, x_j) \in \mathcal{P}\}$ be the vertex set and the edge set, respectively. In the initial graph, each vertex v_i corresponds to an instance $x_i \in \mathcal{X}$. However, all vertices, whose corresponding instances are equal ($\approx$) to each other, are merged into a single vertex. In such a case, incident edges are modified accordingly. Therefore, each vertex represents a set of one or more instances.

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

$$\sigma : \mathcal{V} \to \{1, 2, \ldots, |\mathcal{V}|\} \tag{35}$$

satisfying the constraint

$$\sigma(v_i) < \sigma(v_j) \text{ for all } (v_i, v_j) \in \mathcal{E}. \tag{36}$$

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 σ;

$$\mathcal{L} = \{(x_i, x_j) : \sigma(v_i) \leq \sigma(v_j) \text{ and } x_i, x_j \in \mathcal{X}\} \tag{37}$$

where v_i and v_j are the vertices containing x_i and x_j, respectively. Note that a linearly extended ordering $\mathcal{L}$ is not unique in general. 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, the chainization algorithm [19] uses a pairwise comparator. It has the same architecture as the comparator in OL [24], except that it classifies the ordering between instances x and y into two cases: $x \preccurlyeq y$ or $x \succcurlyeq y$. Chainization first trains the comparator using the known ordered pairs in $\mathcal{P}$. The training loss is given by

$$\ell = [x \not\approx y]\ell_{\text{ce}}(p^{xy}, q^{xy}) + [x \approx y]D(p^{xy} \| q^{xy}) \tag{38}$$

where $[\cdot]$ is the indicator function. Also, $p^{xy} = (p^{xy}_{\preccurlyeq}, p^{xy}_{\succcurlyeq})$ and $q^{xy} = (q^{xy}_{\preccurlyeq}, q^{xy}_{\succcurlyeq})$ are the softmax probabilities yielded by the comparator and ground-truth one-hot vector. If $x \prec y$ or $x \succ y$, they use the cross-entropy loss ℓ_{ce} as in OL [24]. However, if $x \approx y$, they set $q^{xy}_{\preccurlyeq} = q^{xy}_{\succcurlyeq} = 0.5$ and use the KL-divergence D, instead of the cross-entropy.

Algorithm 1 Chainization

Input: Directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$ for $\mathcal{P}$

Train a comparator on $\mathcal{P}$ for warm-up epochs;
repeat
  $\mathcal{Q} \leftarrow \varnothing$; $t \leftarrow 1$;
  Add all vertices $v \in \mathcal{V}$ with $\delta(v) = 0$ to $\mathcal{Q}$;
  while $\mathcal{Q} \neq \varnothing$ **do**
    Remove the optimal v^* in (40) from $\mathcal{Q}$;
    $\sigma(v^*) \leftarrow t$; $t \leftarrow t + 1$;
    for all adjacent vertex w of v^* in G **do**
      Remove edge (v^*, w) from $\mathcal{E}$;
      if $\delta(w) = 0$ **then**
        Add w to $\mathcal{Q}$;
      end if
    end for
  end while
  Obtain a chain from the sorting function σ;
  Shorten it to yield the linear ordering $\mathcal{L}$ in (41);
  Build a set $\mathcal{T}$ of pseudo pairs;
  Fine-tune the comparator on $\mathcal{P} \cup \mathcal{T}$;
until predefined number of epochs;

Output: Linear ordering $\mathcal{L}$, comparator

Chainization: To determine the sorting function σ in (35), Lee and Kim [19] developed the chainization algorithm in Algorithm 1, by extending Kahn's topological sorting algorithm [15]. However, whereas the Kahn's algorithm obtains an arbitrary linear extension of $\mathcal{P}$, chainization yields a meaningful linear ordering by estimating missing ordering information, not included in $\mathcal{P}$, using the pairwise comparator.

Chainization iteratively selects a vertex v from the graph G and appends it to the sorted list. In other words, at iteration t, it selects v and set $\sigma(v) = t$. First, it forms a set $\mathcal{Q} = \{v : \delta(v) = 0 \text{ and } v \in \mathcal{V}\}$, where $\delta(\cdot)$ denotes the indegree of a vertex. Second, it selects 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, the probability that a vertex v precedes another vertex w is defined as

$$p(v, w) = \frac{1}{kl} \sum_{i=1}^{k} \sum_{j=1}^{l} p_{\preccurlyeq}^{x_i y_j} \tag{39}$$

where $v = \{x_1, \ldots, x_k\}$ and $w = \{y_1, \ldots, y_l\}$. It also defines the priority score π of each vertex $v \in \mathcal{Q}$ as $\pi(v) = \sum_{w \in \mathcal{Q}: w \neq v} p(v, w)$. Then, it chooses the

highest-priority vertex

$$v^* = \operatorname*{argmax}_{v \in \mathcal{Q}} \pi(v) \tag{40}$$

and set $\sigma(v^*) = t$. Chainization repeats 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 Figure 7. Let $(w_1, w_2, \ldots, w_{|\mathcal{V}|})$ denote this chain, where $w_i = v_j$ if $\sigma(v_j) = i$. Note that adjacent vertices in the chain may contain instances equal to another, since the graph representation is performed without full annotations of instance equalities ($\approx$).

Therefore, chainization merges vertices in the chain, which likely come from the same underlying class. Specifically, it merges the adjacent vertices w_i and w_{i+1} with the lowest probability $p(w_i, w_{i+1})$ in (39) 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.

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

$$\mathcal{L} = \{(x, y) : x \in w_i, y \in w_j \text{ and } i \leq j\}. \tag{41}$$

Notice that $\mathcal{L}$ is obtained using the output of the comparator in (39), 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 Figure 7, chainization forms a set $\mathcal{T}$ of pseudo training pairs by sampling ordered pairs (x, y), where $x \in w_i$, $y \in w_j$, and $j - i > \tau$, and add them to $\mathcal{T}$. Here, τ is a sampling threshold.

Then, chainization fine-tunes the comparator using the augmented training set $\mathcal{P} \cup \mathcal{T}$ and re-estimates the linear ordering $\mathcal{L}$ with the fine-tuned comparator alternately.

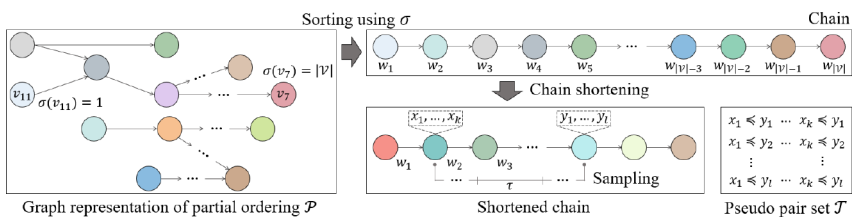


Figure 7: An overview of chainization. This figure is from [19].

5 Applications

In this section, we summarize and compare the results of the order learning algorithms in various applications, including facial age estimation, HCI classification, and aesthetic quality assessment. Note that these order learning algorithms can also be applied to other tasks as well, such as image retrieval, aesthetic image cropping, and medical assessment, but we do not address such tasks in this survey paper.

5.1 Facial Age Estimation

Facial age estimation is a vision task to predict the real or apparent age from a person's facial image. It is one of the most popular rank estimation tasks.

5.1.1 Datasets

MORPH II [33]: It is the most widely used dataset for facial age estimation, containing about 55,000 facial images of 13,617 subjects in the age range [16, 77]. In each image, the gender and race labels are annotated as well. Most rank estimation algorithms employ the four evaluation settings A, B, C, and D.

- Setting A: 5,492 images of Caucasians are sampled and then randomly split into train and test sets with a ratio of 8:2.
- Setting B: About 21K images of Caucasians and Africans are randomly chosen so that the ratio between Caucasians and Africans is 1:1 and that between females and males is 1:3. Then, it is divided into three subsets (S1, S2, S3). The training and testing are repeated twice; 1) training on S1, testing on S2+S3, and 2) training on S2, testing on S1+S3. The average scores of two evaluations are reported.
- Setting C: The whole dataset is randomly divided into five folds with the constraint that images of the same person should belong to only one fold. Then, the 5-fold cross-validation is performed.
- Setting D: The whole dataset is randomly split into five folds without any constraint. Then, the 5-fold cross-validation is performed.

CACD [8]: It contains about 160,000 images of 2,000 celebrities, which are divided into three subsets by celebrities: 1,800 for training, 80 for validation, and 120 for testing. The age range is [14, 62].

UTK [49]: It provides about 20,000 facial images, which are divided into 13,147 for training and 3,287 for testing, in the age range [0, 116].

Adience [21]: It is for age group estimation. It contains 26,580 facial images of 2,284 subjects, which are grouped into 8 classes: 0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, and over 60-year-olds. The 5-fold subject-exclusive (SE) cross-validation evaluation setting is widely used.

IMDB-WIKI [34]: It contains about 500,000 celebrity images, crawled from IMDB and Wikipedia. It is only used for network pre-training [23, 32, 34, 44, 46].

5.1.2 Comparative Assessment

Table 1 compares the rank estimation results on MORPH II. For evaluation, the mean absolute error (MAE) and cumulative score (CS) metrics are adopted. MAE is the average absolute error between predicted and ground-truth ages, and CS is the percentage of test instances whose absolute errors are less than or equal to 5. Note that all algorithms in Table 1 employ VGG16 as the encoder backbones, except for C3AE using a shallow CNN.

In most tests, order learning algorithms outperform other age estimators, indicating that order learning is effective for age estimation. Non-order-learning algorithms BridgeNet and AVDL — which are based on ordinal regression — show good results. However, DRC-ORID, MWR, and GOL achieve better or comparative scores than these non-order-learning algorithms.

Table 1: Comparison of facial age estimation results in the four evaluation settings (A, B, C, and D) of MORPH II. Here, * means that IMDB-WIKI pre-training is performed.

	Setting A		Setting B		Setting C		Setting D	
Algorithm	MAE	CS(%)	MAE	CS(%)	MAE	CS(%)	MAE	CS(%)
OR-CNN [30]	-	-	-	-	-	-	3.27	73.0
Tan *et al.* [50]	-	-	3.03	-	-	-	-	-
Ranking-CNN [9]	-	-	-	-	-	-	2.96	85.0
DEX [34]*	2.68	-	-	-	-	-	-	-
DMTL [14]	-	-	-	-	3.00	85.3	-	-
CMT [48]	-	-	-	-	2.91	-	-	-
DRFs [39]	2.91	82.9	2.98	-	-	-	2.17	91.3
AGEn [44]*	2.52	85.0	2.70	83.0	-	-	-	-
MV [32]*	-	-	-	-	2.79	-	2.16	-
C3AE [7]*	-	-	-	-	-	-	2.75	-
BridgeNet [23]*	2.38	91.0	2.63	86.0	-	-	-	-
AVDL [46]*	2.37	-	2.53	-	-	-	**1.94**	-
OL [24]*	2.41	91.7	2.75	88.2	2.68	88.8	2.22	93.3
DRC-ORID [18]*	2.26	**93.8**	**2.51**	89.7	2.58	89.5	2.16	93.5
MWR [41]*	2.24	93.5	2.55	**90.1**	2.61	89.5	2.16	93.0
GOL [20]	**2.17**	**93.8**	2.60	89.3	**2.51**	**90.0**	2.09	**94.2**

Table 2: Comparison in the train and validation settings of CACD and also on UTK and Adience. OL and DRC-ORID do not provide their results on these datasets.

	Train	Validation	UTK	Adience	
Algorithm	MAE	MAE	MAE	Acc.	MAE
OR-CNN [30]	-	-	-	56.7	0.54
dLDLF [40]	4.73	6.77	-	-	-
AGEn [44]	4.68	-	-	-	-
CNNPOR [26]	-	-	-	57.4	0.55
DRFs [39]	4.64	5.77	-	-	-
GP-DNNOR [28]	-	-	-	57.4	0.54
SORD [10]	-	-	-	59.6	0.49
CORAL [3]	-	-	5.47	-	-
Gustafsson *et al.* [12]	-	-	4.65	-	-
POE [22]	-	-	-	60.5	0.47
Berg *et al.* [1]	-	-	4.55	-	-
MWR [41]	4.76	5.75	4.49	62.2	0.46
GOL [20]	**4.52**	**5.58**	**4.35**	**62.5**	**0.43**

Next, Table 2 compares the performances on the CACD, UTK, and Adience datasets. The order learning algorithms provide decent results on these datasets as well. Note that [12] and [1] adopt the deeper ResNet50 as their encoder backbones, while the order learning algorithms employ VGG16. Nevertheless, both MWR and GOL outperform them on the UTK dataset. Also, the order learning algorithms outperform the other algorithms with a large performance gap on the Adience dataset.

Table 4 compares the complexities of some age estimators. We see that the order learning algorithms demand fewer parameters than the conventional non-order-learning algorithms DEX and MV. Even with relatively lightweight architecture, the order learning algorithms estimate the ranks more effectively.

5.2 HCI Classifiacation

HCI is a task to determine the decade when a photograph was taken. The HCI dataset [31] contains images from five decades 1930s $\sim$ 1970s. There are 265 images in each decade. These images are randomly split into three subsets: 210 for training, 5 for validation, and 50 for testing. Then, the 10-fold cross-validation is performed.

Table 3 shows the results on the HCI dataset. The order learning algorithms [20, 41] outperform the conventional rank estimators. This indicates

Table 3: Accuracy (%) and MAE comparison on the HCI dataset. OL does not provide its results on this dataset.

Algorithm	Acc.	MAE
Frank & Hall [11]	41.4	0.99
Cardoso *et al.* [4]	41.3	0.95
Palermo *et al.* [31]	44.9	0.93
RED-SVM [25]	35.9	0.96
Martin *et al.* [29]	42.8	0.87
OR-CNN [30]	38.7	0.95
CNNPOR [26]	50.1	0.82
GP-DNNOR [28]	46.6	0.76
POE [22]	<u>54.7</u>	0.66
DRC-ORID [18]	44.7	0.80
MWR [41]	52.2	<u>0.60</u>
GOL [20]	**56.2**	**0.55**

Table 4: Comparison of model complexities.

Algorithm	DEX	MV	OL	DRC-ORID	MWR-G	GOL	Chainization
Parameters (M)	138	138	15.51	45.86	15.77	14.75	15.51

that the order learning algorithms perform well even when only a small number of training images are available.

5.3 Aesthetic Quality Assessment

Aesthetic quality assessment aims to estimate the aesthetic score of an image. It is challenging because aesthetic criteria are subjective and ambiguous.

5.3.1 Datasets

Aesthetics [37]: It contains about 15,687 Flickr images in four categories: nature, animal, urban, and people. Each image was scored from 1 to 5 by more than five annotators. The median score is regarded as the ground-truth. As in [10], we randomly divide the whole dataset into three subsets: 75% for training, 5% for validation, and 20% for testing. Then, the 5-fold cross-validation is performed.

AADB [16]: It provides 10,000 photographs of various themes such as scenery and close-up. It is divided into 8,500 training, 500 validation, and

Table 5: Comparison of aesthetic quality assessment results on the Aesthetics dataset.

	Nature		Animal		Urban		People		Overall	
Algorithm	Accuracy (%)	MAE	Accuracy (%)	MAE	Accuracy (%)	MAE	Accuracy (%)	MAE	Accuracy (%)	MAE
OR-CNN [30]	69.8	0.31	69.1	0.33	66.5	0.35	70.4	0.31	69.0	0.33
CNNPOR [26]	71.9	0.29	69.3	0.32	69.1	0.33	69.9	0.32	70.1	0.32
SORD [10]	73.6	**0.27**	70.3	0.31	73.3	0.28	70.6	0.31	72.0	0.29
POE *et al.* [22]	73.6	**0.27**	71.1	0.30	72.8	0.28	**72.2**	**0.29**	72.4	0.29
GOL [20]	**73.8**	**0.27**	**72.4**	**0.28**	**74.2**	**0.26**	69.6	0.31	**72.7**	**0.28**

Table 6: Comparison of aesthetic quality assessment results on the AADB dataset.

Algorithm	Reg-Net	ASM	DRC-ORID
MAE	0.1268	0.1141	**0.1056**

1,000 test images. Each image is annotated with an aesthetic score in $[0, 1]$. These continuous scores are quantized with a step size of 0.01.

5.3.2 Comparative Assessment

Table 5 compares the performances on the Aesthetics dataset. GOL, an order learning algorithm, outperforms all conventional algorithms. This indicates that order learning performs reliably in aesthetic assessment as well.

Table 6 compares the results on the AADB dataset. ASM [17] also adopts pairwise comparison as order learning algorithms. However, DRC-ORID achieves better results by finding optimal references for each test instance.

5.4 Weakly Supervised Order Learning

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, the supervised learning scenario corresponds to $\gamma = 100\%$. However, at a lower γ, the amount of information available for training is reduced and the training becomes more difficult.

Figure 8 compares the results of chainization [19] and OL [24] on the Adience dataset. Both chainization and OL need reference instances with known ranks to estimate the rank of an instance. Thus, for each rank, an instance is randomly selected from the training set as a reference. At all γ's, chainization outperforms OL. Especially, at a low $\gamma = 0.005\%$, OL fails to obtain a reliable comparator due to the lack of training pairs, whereas chainization achieves a much higher accuracy by optimizing the comparator

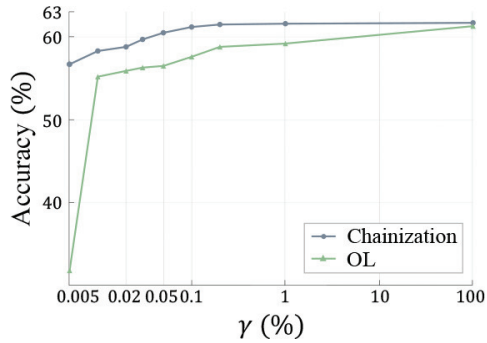


Figure 8: Comparison of chainization with OL on the Adience dataset. The x-axis is in a logarithmic scale.

Table 7: Comparison of rank estimation results on the Adience dataset.

Algorithm	Accuracy (%)	MAE
OR-CNN [30]	56.7 ± 6.0	0.54 ± 0.08
CNNPOR [26]	57.4 ± 5.8	0.55 ± 0.08
GP-DNNOR [28]	57.4 ± 5.5	0.54 ± 0.07
SORD [10]	59.6 ± 3.6	0.49 ± 0.05
POE [22]	**60.5 ± 4.4**	0.47 ± 0.06
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

over pseudo pairs. It demonstrates that order learning performance can be improved via chainization under weakly supervised scenarios.

Table 7 compares results on the Adience dataset. Even with weak supervision, the order learning performances of chainization are competitive. For example, using only 0.03% of the ordering relations in $\mathcal{L}$, chainization performs better than the others, except for POE [22]. Moreover, at $\gamma = 0.08\%$, it reaches the performances of POE. This confirms that order learning can be effectively applied to weakly supervised scenarios as well.

6 Future Work

The order learning algorithms have shown promising results in various rank estimation tasks. However, there is still plenty of room for improvement.

First, these algorithms select a threshold τ heuristically. Note that τ in (1) determines the ordering relationship between instances. Even though a few schemes for selecting τ have been analyzed in [24, 41], these schemes require multiple experiments, which may be too demanding, to find an optimal τ. Hence, a more systematic way to select an optimal τ should be developed for wider applications of order learning. Second, an optimal chain division scheme should be developed. Even for human beings, not every comparison is easy. Therefore, it is important to discover optimal chains, so instances in each chain are easily compared with one another. To this end, DRC-ORID [18] was proposed, but it is inefficient and may not be optimal because it performs chain clustering and network training separately. One possible way to handle this issue may be to combine GOL and DRC-ORID so as to enforce instances to form parallel chains in the embedding space.

7 Conclusions

In this overview, we first provided a survey of the order learning algorithms for rank estimation. First, we introduced conventional rank estimation algorithms relevant to order learning. Then, we described the state-of-the-art order learning algorithms in detail. Finally, we reported the results of order learning on three representative rank estimation tasks — facial age estimation, HCI classification, and aesthetic quality assessment — and demonstrated that order learning performs excellently on these tasks.

References

[1] A. Berg, M. Oskarsson, and M. O'Connor, "Deep ordinal regression with label diversity," in *ICPR*, 2021.

[2] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, 1996.

[3] W. Cao, V. Mirjalili, and S. Raschka, "Rank-consistent ordinal regression for neural networks," *Pattern Recog. Lett.*, 140, 2020, 325–31.

[4] J. S. Cardoso and J. F. P. da Costa, "Learning to classify ordinal data: The data replication method," *Journal of Machine Learning Research*, 8, 2007, 1393–429.

[5] K.-Y. Chang and C.-S. Chen, "A learning framework for age rank estimation based on face images with scattering transform," *IEEE Trans. Image Process.*, 24(3), 2015, 785–98.

[6] K.-Y. Chang, C.-S. Chen, and Y.-P. Hung, "A ranking approach for human age estimation based on face images," in *ICPR*, 2010.

[7] Z. Chao, S. Liu, X. Xu, and C. Zhu, "C3AE: Exploring the limits of compact model for age estimation," in *CVPR*, 2019.
[8] B.-C. Chen, C.-S. Chen, and W. H. Hsu, "Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset," *IEEE Trans. Multimedia*, 17, 2015, 804–15.
[9] S. Chen, C. Zhang, M. Dong, J. Le, and M. Rao, "Using ranking-CNN for age estimation," in *CVPR*, 2017.
[10] R. Diaz and A. Marathe, "Soft labels for ordinal regression," in *CVPR*, 2019.
[11] E. Frank and M. Hall, "A simple approach to ordinal classification," in *ECML-PKDD*, 2001.
[12] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schon, "Energy-based models for deep probabilistic regression," in *ECCV*, 2020.
[13] P. A. Gutiérrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas-Martinez, "Ordinal regression methods: Survey and experimental study," *IEEE Trans. Knowl. Data Eng.*, 28(1), 2015, 127–46.
[14] H. Hu, A. K. Jain, F. Wang, S. Shan, and X. Chen, "Heterogeneous face attribute estimation: A deep multi-task learning approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, 40, 2017, 2597–609.
[15] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, 5(11), 1962, 558–62.
[16] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes, "Photo aesthetics ranking network with attributes and content adaptation," in *ECCV*, 2016.
[17] J.-T. Lee and C.-S. Kim, "Image Aesthetic Assessment Based on Pairwise Comparison – A Unified Approach to Score Regression, Binary Classification, and Personalization," in *ICCV*, 2019.
[18] S.-H. Lee and C.-S. Kim, "Deep Repulsive Clustering of Ordered Data Based on Order-Identitiy Decomposition," in *ICLR*, 2021.
[19] S.-H. Lee and C.-S. Kim, "Order Learning Using Partially Ordered Data via Chainization," in *ECCV*, 2022.
[20] S.-H. Lee, N.-H. Shin, and C.-S. Kim, "Geometric Order Learning for Rank Estimation," in *NeurIPS*, 2022.
[21] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *CVPR Workshops*, 2015.
[22] W. Li, X. Huang, J. Lu, J. Feng, and J. Zhou, "Learning Probabilistic Ordinal Embeddings for Uncertainty-Aware Regression," in *CVPR*, 2021.
[23] W. Li, J. Lu, J. Feng, C. Xu, J. Zhou, and Q. Tian, "BridgeNet: A continuity-aware probabilistic network for age estimation," in *CVPR*, 2019.
[24] K. Lim, N.-H. Shin, Y.-Y. Lee, and C.-S. Kim, "Order learning and its application to age estimation," in *ICLR*, 2020.

[25] H.-T. Lin and L. Li, "Reduction from cost-sensitive ordinal ranking to weighted binary classification," *Neural Computation*, 24(5), 2012, 1329–67.
[26] Y. Liu, A. W. K. Kong, and C. K. Goh, "A constrained deep neural network for ordinal regression," in *CVPR*, 2018.
[27] Y. Liu, A. Wai Kin Kong, and C. Keong Goh, "Deep ordinal regression based on data relationship for small datasets," in *IJCAI*, 2017.
[28] Y. Liu, F. Wang, and A. W. K. Kong, "Probabilistic deep ordinal regression based on Gaussian processes," in *CVPR*, 2019.
[29] P. Martin, A. Doucet, and F. Jurie, "Dating color images with ordinal classification," in *Proc. ACM ICMR*, 2014.
[30] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, "Ordinal regression with multiple output CNN for age estimation," in *CVPR*, 2016.
[31] F. Palermo, J. Hays, and A. A. Efros, "Dating historical color images," in *ECCV*, 2012.
[32] H. Pan, H. Han, S. Shan, and X. Chen, "Mean-variance loss for deep age estimation from a face," in *CVPR*, 2018.
[33] K. Ricanek and T. Tesafaye, "MORPH: A longitudinal image database of normal adult age-progression," in *FGR*, 2006.
[34] R. Rothe, R. Timofte, and L. V. Gul, "Deep expectation of real and apparent age from a single image without facial landmarks," *Int. J. Comput. Vis.*, 126(2-4), 2018, 144–57.
[35] R. Rothe, R. Timofte, and L. V. Gul, "DEX: Deep expectation of apparent age from a single image," in *ICCV Workshops*, 2015.
[36] T. L. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, 15(3), 1977, 234–81.
[37] R. Schifanella, M. Redi, and L. M. Aiello, "An image is worth more than a thousand favorites: Surfacing the hidden beauty of Flickr pictures," in *ICWSM*, 2015.
[38] B. S. W. Schröder, *Ordered Sets: An Introduction*, Springer, 2003.
[39] W. Shen, Y. Guo, Y. Wang, K. Zhao, B. Wang, and A. Yuille, "Deep regression forests for age estimation," in *CVPR*, 2018.
[40] W. Shen, K. Zhao, Y. Guo, and A. Yuille, "Label distribution learning forests," in *NeurIPS*, 2017.
[41] N.-H. Shin, S.-H. Lee, and C.-S. Kim, "Moving window regression: A novel approach to ordinal regression," in *CVPR*, 2022.
[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICML*, 2015.
[43] Y. Souri, E. Noury, and E. Adeli, "Deep relative attributes," in *ACCV*, 2016.
[44] Z. Tan, J. Wan, Z. Lei, R. Zhi, G. Guo, and S. Z. Li, "Efficient group-n encoding and decoding for facial age estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(11), 2017, 2610–23.

[45] G. Tutz, "Ordinal regression: A review and a taxonomy of models," *Wiley Interdisciplinary Reviews: Computational Statistics*, 14(2), 2022.
[46] X. Wen, B. Li, H. Guo, Z. Liu, G. Hu, M. Tang, and J. Wang, "Adaptive variance based label distribution learning for facial age estimation," in *ECCV*, 2020.
[47] D. Yi, Z. Lei, and S. Z. Li, "Age estimation by multi-scale convolutional network," in *ACCV*, 2014.
[48] B. Yoo, Y. Kwak, Y. Kim, C. Choi, and J. Kim, "Deep facial age estimation using conditional multitask learning with weak label expansion," *IEEE Signal Process. Lett.*, 25, 2018, 808–12.
[49] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *CVPR*, 2017.
[50] T. Zichang, S. Zhou, J. Wan, Z. Lei, and S. Z. Li, "Age estimation based on a single network with soft softmax of aging modeling," in *ACCV*, 2016.