Deep Ordinal Regression Based on Data Relationship for Small Datasets

Yanzhu Liu[1], **Adams Wai Kin Kong**[1], **Chi Keong Goh**[2]
Nanyang Technological University, Singapore[1]
Rolls-Royce Advanced Technology Centre, Singapore[2]
liuy0109@e.ntu.edu.sg, adamskong@ntu.edu.sg, ChiKeong.Goh@Rolls-Royce.com

Abstract

Ordinal regression aims to classify instances into ordinal categories. As with other supervised learning problems, learning an effective deep ordinal model from a small dataset is challenging. This paper proposes a new approach which transforms the ordinal regression problem to binary classification problems and uses triplets with instances from different categories to train deep neural networks such that high-level features describing their ordinal relationship can be extracted automatically. In the testing phase, triplets are formed by a testing instance and other instances with known ranks. A decoder is designed to estimate the rank of the testing instance based on the outputs of the network. Because of the data argumentation by permutation, deep learning can work for ordinal regression even on small datasets. Experimental results on the historical color image benchmark and MSRA image search datasets demonstrate that the proposed algorithm outperforms the traditional deep learning approach and is comparable with other state-of-the-art methods, which are highly based on prior knowledge to design effective features.

1 Introduction

The ordinal regression is a supervised learning problem, which aims at predicting the rank label of an input vector. The natural order of the labels (e.g., 1,2,3) indicates the order of the ranks. This problem is different from the multi-class classification problem as there is an ordinal relationship between the categories. It is also different from the metric regression problem because the target values to be predicted are discrete and the distances between different categories are not defined.

From the problem setting, obviously, the most important task of ordinal regression methods is to model the ordinal relationship between the categories. Recently, a number of machine learning approaches have been proposed for ordinal regression. The main difference between these approaches is the way to make use of the ordinal information. For example, the max-margin based approaches [Shashua and Levin, 2002][Chu and Keerthi, 2007] predict contiguous boundaries to split the ordinal classes and the projection based approaches [Liu *et al.*, 2011][Tian and Chen, 2015] map instances to a new space preserving the ordinal relationship between classes. However, few of existing works extract and represent the ordinal relationship from instances directly. Herbrich et al. (1999) [Herbrich *et al.*, 1999] proposed to explore the relationship between instances by using feature pairs and the ordinal relationship is represented by whose rank is greater in a pair. However, this pairwise representation cannot describe the ordinal relationship among instances completely. For example, it cannot distinguish the difference between pairs of instances from rank 1 and rank 2 and those from rank 1 and rank 3. Another limitation of this method [Herbrich *et al.*, 1999] is that the feature pairs are constructed highly based on prior knowledge. The strategy of the pairwise feature construction greatly influences the performance of the method and thus it is task dependent.

Being able to automatically extract high-level features from raw data, deep neural networks (DNNs), such as convolutional neural networks (CNNs), have attracted great attention in these several years and performed very well on many classification problems. However, instances are inputted individually into DNNs. To our best knowledge, there is no any existing work employing DNNs to extract and represent high-level features describing relationship among data instances for ordinal regression problems. Generally speaking, to train a deep neural network, a large training dataset is necessary. Using deep learning on small datasets is challenging, but many real-world ordinal regression problems are in fact small data problems. For example, in computer-aided diagnostic problems, datasets of rare disease grading and tumor staging are usually small as collecting data for these diseases is difficult, expensive and invasive. Even for more common diseases, the datasets are not large, because of privacy. In this paper, representing instances relationship and small dataset problems are two challenges that we are targeting.

This paper proposes to use triplets whose elements are from different ranks as samples to explore the ordinal data relationship. The intuition is simple: if a method can predict that the rank of an input x is greater than $k-1$ and smaller than $k+1$, then the rank label of x will be k. Therefore, the proposed approach transforms a m-rank ordinal regression problem to m binary classification problems with triplets as inputs. And the k-th classifier answers the question: "Is the rank of x greater than $k-1$ and smaller than

$k+1$?" One of state-of-the-art ordinal regression methods RED-SVM [Lin and Li, 2012] trained a binary classifier to answer "Is the rank of x greater than k or not?" Comparing with RED-SVM, the proposed question in this paper is more precise so that it is straightforward to recover the rank label from the answers. In the proposed approach, DNNs are adopted to extract high-level features from the triplets, and for each rank k, a separate DNN is trained for the corresponding binary classification problem. Because the distance between every two adjacent ranks can be different, separate DNNs are used instead of one multi-class CNN. A significant benefit of the proposed approach is data augmentation. If the size of a training dataset is n, the number of triplets used for training the CNNs will be $O(n^3)$. Therefore, the proposed approach makes deep learning on small datasets possible.

The rest of this paper is organized as follows. Section 2 reviews the literature of ordinal regression. Section 3 describes the proposed deep ordinal regression framework. Section 4 reports the experimental results. Section 5 gives some conclusive remarks.

2 Related Work

Comprehensive reviews of ordinal regression can be found in [McCullagh and Nelder, 1989], [O'Connell, 2006] and [Gutierrez *et al.*, 2016]. Gutierrez et al.'s survey [Gutierrez *et al.*, 2016] is the most recent one, which classifies ordinal regression approaches into three categories: naïve, binary decomposition and threshold approaches. The naïve approaches adapt nominal classification or metric regression methods to solve the ordinal regression problem. Binary decomposition separates the ordinal target labels to binary ones, which are then estimated by a single or multiple models [Herbrich *et al.*, 1999][Frank and Hall, 2001]. Threshold approaches assume that there is a latent function mapping the instances to a real line, and the categories of the instances are intervals on the line. The natural order of interval boundaries on the real line represents the ordinal relationship between categories [Chu and Keerthi, 2007][Chu and Ghahramani, 2005].

However, from the data point of view, most of existing approaches are pointwise. Their input spaces are the raw feature spaces and they estimate weights or parameters for instances inputted separately instead of pairs or lists of instances. For example, SVOR [Chu and Keerthi, 2007], a SVM based method, estimates the weight w for input vectors x and boundaries b, and the decision criteria is that the rank of x is k if and only if $w^T x \in [b_{k-1}, b_k]$, where b_k is the boundary separating rank k and rank $k+1$. A neural network based approach [Cheng *et al.*, 2008] employed a standard feedforward neural network with m outputs for the m-rank ordinal regression problem. Also the network weights w are estimated for input vectors x and the decision function $f(x) = \sum_{j=1}^{H} \beta_j B_j(x, w_j)$, where H is the number of hidden nodes, w_j is the input weight vector for the j-th hidden node, $B_j(x, w_j)$ is a basis function, and β_j is the j-th output weight. A probabilistic method GPOR [Chu and Ghahramani, 2005] uses Gaussian Process to estimate the parameters θ for the conditional probability $p(k|x, D, \theta)$, where k is a rank label and D is a training set. Then the rank of x is predicted as k with the maximum conditional probability. In all above pointwise methods, the parameters are learned from individual data points, and therefore, the relationship between instances is not explored explicitly. The method proposed by Herbrich et al. (1999) [Herbrich *et al.*, 1999] can be viewed as a pairwise method. Input vectors are paired up first, for example (x_i, x_j), and then their difference $x_i - x_j$ as a feature vector is inputted into a standard SVM and the label is $sign(y_i - y_j)$, where y_i is the rank of x_i and $sign(\cdot)$ is a sign function. However, to form the feature vector for a pair by subtracting one element from another is ad-hoc. And $sign(\cdot)$ operation is only able to indicate whose rank is greater, but cannot show how far between the ranks of the two elements. Furthermore, they rely on prior knowledge to design effective features.

In the last decade, although deep learning has achieved great success on classification, there are very few works to adopt DNNs to ordinal regression problems. Niu et al. (2016) [Niu *et al.*, 2016] have recently adopted CNN for age estimation and claimed that they are the first one to address ordinal regression problems by using CNN. Their method is a pointwise approach. The training instances are inputted to CNN individually and the relationships between input instances are not considered. And as traditional deep learning methods, their method is more applicable for large scale datasets.

3 A Convolutional Neural Network for Ordinal Regression

An ordinal regression problem with m ranks denoted by $Y = \{1, 2, \cdots, m\}$ is considered, where the natural order of the numbers in Y indicates the order of the ranks. A training set with labeled instances $T = \{(x_i, y_i) | x_i \in X, y_i \in Y\}$ is given, where X is the input space. The target is to predict the rank $y_t \in Y$ of an input $x_t \in X$. In the rest of this section, the outline of the proposed approach will be provided first, and then its key components, including a pre-trained CNN and a decoding scheme will be presented.

3.1 The Proposed Approach

The proposed deep neural network for ordinal regression is to predict rank label by answering the questions: "Is $k-1 < y_t < k+1$ true?" for all $k \in Y$. Obviously, if the answer is "yes" for a certain k, then the predicted rank label y_t will be k. In the proposed approach, for each rank k in Y, a separate binary classifier is trained to answer the above question. Since convolutional neural networks are used in the current implementation, the proposed method is named convolutional neural network for ordinal regression (CNNOR). Algorithm 1 gives the pseudo code of the CNNOR. It consists of three steps: pre-training (line 1), training (line 2-6) and decoding (line 7-11). Given a training set T with labeled instances and a testing point x_t, the goal of CNNOR is to predict the rank of x_t being most likely between which two ranks. In other words, CNNOR aims to predict whether a triplet (x_i, x_t, x_j) is consistent in order, where "consistent" means $x_i \in X_{k-1}$, $x_t \in X_k$, and $x_j \in X_{k+1}$. Therefore, Algorithm 1 starts from training CNN_{reg} which aims to make the ranks of x_i and x_j in order first for all possible triplets (x_i, x_t, x_j).

Algorithm 1 Pseudo code of CNNOR

Input: Training set $T = \{(x_i, y_i)|x_i \in X, y_i \in Y\}$ where X is an input space, $Y = \{1, 2, \cdots, m\}$, and a test point $x_t \in X$.
Output: y_t, the predicted rank of x_t.

1: Train CNN$_{reg}$ on T by minimizing Eq. 1.
2: Split T into m subsets $X_1, \cdots, X_m$ with $X_k = \{x_i|(x_i, y_i) \in T, y_i = k\}$ for $k = 1, \cdots, m$.
3: **for** $k = 1$ to m **do**
4: Construct positive and negative triplet sets D_k^+ and D_k^- using Eq. 2. (Denote $D_k = D_k^+ \cup D_k^-$)
5: Train CNN$_k$ on D_k with weights initialized from the pre-trained CNN$_{reg}$ model.
6: **end for**
7: **for** $k = 1$ to m **do**
8: Construct d test triplets D_k^t by Eq. 3.
9: Input D_k^t to CNN$_k$ and assign $p_k = \frac{c_k}{d}$ where c_k is the number of positive predictions for rank k.
10: **end for**
11: **return** $y_t = argmax_k(p_k)$.

Let $\phi(x)$ denote the output of the network CNN$_{reg}$ which is an one-dimensional real value, and x be an instance. The objective of CNN$_{reg}$ is to learn a function $\phi(\cdot)$ which maps x_i and x_j to a real line such that $\phi(x_i) < \phi(x_j)$ if $y_i < y_j$. Hence, we construct the training set with lists $(x_i^1, x_i^2, \cdots, x_i^k, \cdots, x_i^m)$ as instances, where x_i^k is from rank k. The loss function is defined by Eq. 1,

$$\sum_{i=1}^{n} \sum_{k=1}^{m-1} max(0, g + w \cdot \phi(x_i^k) - w \cdot \phi(x_i^{k+1})) \quad (1)$$

where x_i^k is from the k-th rank, m is the number of ranks and n is the size of training set (i.e., the number of lists in the training set), g is a hyperparameter which controls the margin of mapped value between adjacent ranks. Eq. 1 is named as Ordinal Loss, which calculates the total error of all pairs of instances if their orders are incorrect or their margin is smaller than g. Figure 1 illustrates the Ordinal Loss for a 3-rank ordinal regression problem. The axis is the real line which is the range of $\phi(\cdot)$. The red circles are the set of points mapped from instances of the rank 1, i.e., $\{\phi(x_i)|y_i = 1\}$. The green crosses are the mapped points from the rank 2, and the blue stars are those from the rank 3. g in Eq. 1 is the expected margin between two adjacent ranks as shown in Figure 1. The Ordinal Loss only counts the errors from pairs of adjacent ranks, e.g., e_1, e_2 and e_3. But other errors such as e_4 are not considered explicitly, because if both $(\phi(x_1), \phi(x_2))$ and $(\phi(x_2), \phi(x_3)))$ are in order, it can be inferred that $(\phi(x_1), \phi(x_3)))$ are also in order. In other words, when e_2 is minimized, e_4 is minimized implicitly.

Then, the next step of Algorithm 1 is to learn a separate CNN for each rank to extract high level features describing ordinal relationship. For each rank k, a new training set with triplets as instances is derived. In line 2 of Algorithm 1, the training set T is split into m subsets $X_1, \cdots, X_m$ according

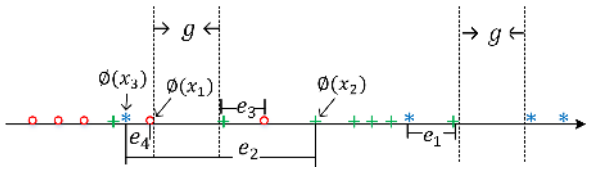


Figure 1: Illustration of Ordinal Loss for a 3-rank problem.

to the rank labels. All the instances in the subset X_k have the same rank label which is k. The new training set D_k is derived as Eq. 2:

$$D_k = D_k^+ \cup D_k^-$$

$$D_k^+ = \begin{cases} \{((x_p, x_j), +1)|x_p \in X_1, x_j \in X_2\} & \text{if } k = 1 \\ \{((x_i, x_p), +1)|x_i \in X_{m-1}, x_p \in X_m\} & \text{if } k = m \\ \{((x_i, x_p, x_j), +1)|x_i \in X_{k-1}, x_p \in X_k, x_j \in X_{k+1}\} & \text{otherwise} \end{cases}$$

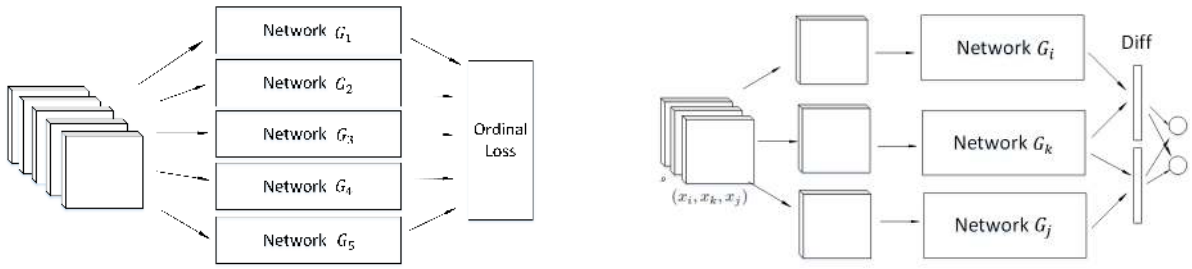
$$D_k^- = \begin{cases} \{((x_p, x_j), -1)|x_p \in X_r, 1 < r \leqslant m, x_j \in X_2\} & \text{if } k = 1 \\ \{((x_i, x_p), -1)|x_p \in X_r, 1 \leqslant r < m, x_i \in X_{m-1}\} & \text{if } k = m \\ \{((x_i, x_p, x_j), -1)|x_i \in X_{k-1}, x_j \in X_{k+1}, x_p \in X_r, r \neq k\} & \text{otherwise} \end{cases} \quad (2)$$

D_k^+ is a positive training set which includes triplets with three elements from ranks $k-1$, k and $k+1$ and the negative training set D_k^- includes the triplets whose middle elements are from other ranks and the other two elements are from the ranks $k-1$ and $k+1$. Because there is no previous rank for rank 1, the training set is formed by pairs (x_p, x_j) where $x_j \in X_2$. The classifier for rank 1 aims to decide whether the rank of x_i is smaller than x_j. Similarly, the training samples for rank m is the pairs (x_i, x_p) where $x_i \in X_{m-1}$. Based on the derived training set, CNN$_k$ is fine-tuned from the pre-trained CNN$_{reg}$ in line 5 of Algorithm 1. Each CNN$_k$ has a binary output, indicating whether the input is consistent in order or not.

In the testing phase, as shown in line 7-11, given a testing point x_t, the test triplets are formed for each CNN$_k$ according to Eq. 3.

$$D_k^t = \begin{cases} \{(x_t, x_j)|x_j \in X_2\} & \text{if } k = 1 \\ \{(x_i, x_t)|x_i \in X_{m-1}\} & \text{if } k = m \\ \{(x_i, x_t, x_j)|x_i \in X_{k-1}, x_j \in X_{k+1}\} & \text{otherwise} \end{cases} \quad (3)$$

For each rank k, d pairs of x_i and x_j are randomly selected from X_{k-1} and X_{k+1} and they are constructed as triplets with the testing point x_t. The decision for the rank label of x_t i.e, y_t is made by majority voting. Each classifier k predicts how many triplets (c_k) of the d inputs are consistent in order respect to rank k, and the percentage $\frac{c_k}{d}$ indicates the probability that x_t belongs to rank k. Finally, the prediction of y_t is assigned to the rank k with maximum probability. It should be pointed out that the proposed approach produces a set of



(a) The architecture of CNN$_{reg}$ (b) The architecture of CNN$_k$

Figure 2: The architecture of CNNOR for a 5-rank problem.

testing instances for one testing point. Through reusing training data, it increases both training and testing data to overcome the weaknesses of traditional deep learning for small dataset problems. Increasing the size of the testing set is not always an issue because of the advancement of hardware and no real-time requirements in some applications e.g., healthcare.

3.2 The Architecture of CNNOR

For a m-rank ordinal regression problem, CNNOR includes one CNN$_{reg}$ and m CNNs, each of which is a binary classifier for one rank. Figure 2 shows the architecture of CNN$_{reg}$ for a 5-rank ordinal regression problem and the network CNN$_k$ for the rank k. The input of CNN$_{reg}$ is a list $(x_1, \cdots, x_m)$ which consists of m images for m-rank ordinal regression problem. Each image x_i in $(x_1, \cdots, x_m)$ is inputted into one of the networks, and all the m networks share the same weights. All the networks in CNN$_{reg}$ have one output neuron, and the output value of the i-th network represents $\phi(x_i)$. All the m outputs are inputted to the Ordinal Loss layer to minimize the loss function in Eq. 1. At each iterations of training, the error of the loss layer is back propagated to all networks in CNN$_{reg}$.

Once the well-trained CNN$_{reg}$ is obtained, its weights are used to initialize the weights of the networks in CNN$_k$. As shown in Figure 2b, the input of CNN$_k$ is a triplet (x_i, x_k, x_j), which consists of three images. Each of the three images is inputted to one of the networks denoted as G_i, G_k and G_j in Figure 2b, and all the three networks share the same weights. CNN$_k$ with $k = 1$ ($k = m$) in Figure 2b only has G_k and G_j (G_i). The layers and the parameters of the networks in CNN$_k$ are same as those in CNN$_{reg}$ except for the last fully-connected layer. The last fully-connected layer of CNN$_{reg}$ has only one output neuron, but it is removed in CNN$_k$ and the features extracted from the previous fully-connected layer are passed to the Diff layer as shown Figure 2b. Because CNN$_k$ is trained to model the rank order of (x_i, x_k, x_j) by exploring the data relationship, the Diff layer is used to represent the ordinal information within triplets. Based on the assumption that the distance between instances in the mapped feature space indicates the distance between their ranks, the Diff layer combines two parts of features $\psi(x_k) - \psi(x_i)$ and $\psi(x_j) - \psi(x_k)$, where $\psi(x_i)$ is the feature vectors exacted from the last fully-connected layers of G_i. It can be concluded from the proposed architecture, although the augmented dataset has n^3 triplets, CNNOR is computational feasible even for large datasets. As shown in Figure 2b, the elements of a triplet are processed individually before the Diff layer. For all (x_i, x_j, x_k), only unique x_i, x_j and x_k are necessary to be computed. And G_i, G_k and G_j share weights. Therefore, before the Diff layer, the computation cost of CNN$_k$ is same as a standard CNN. The operation of the Diff layer is a simple subtraction, which is not an issue for a modern hardware even for huge data.

3.3 The Decoder Based on Majority Voting

A simple decoder is designed to predict the rank label of a testing point x_t from the outputs of CNN$_k$ as shown in line 7-11 of Algorithm 1. Table 1 is the coding matrix of CNNOR for a 5-rank example. Each row is for one rank and each column is for one CNN$_k$. The elements in the matrix represent the training targets for different CNN$_k$ and different ranks. For example, the first column of Table 1 is labeled as $(x_t, 2)$, which represents a testing triplet constructed as Eq. 3 for CNN$_1$, and rank 1 is considered as a positive rank and the rest are negative ranks. In the testing phase, d triplets for each column are predicted and the rank with maximum positive predications is regarded as the rank label of x_t.

4 Evaluation

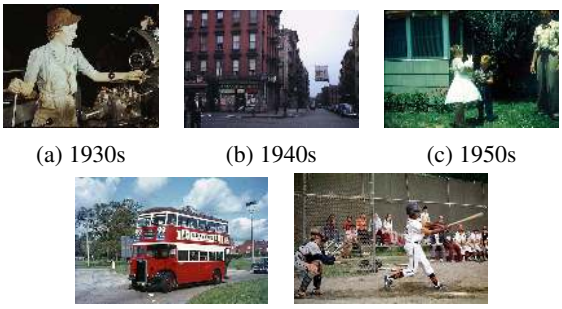
The proposed CNNOR framework is evaluated on a historical color image dataset [Palermo *et al.*, 2012] and an image retrieval dataset MSRA-MM1.0 [Wang *et al.*, 2009]. Two metrics are used as performance indexes. The first one is accuracy defined by $acc = \frac{1}{|T|}\sum_{x_t \in T}[\hat{y}_t = y_t]$, where T is a testing set, $|T|$ is its size, y_t is the ground truth of x_t, $\hat{y}_t$ is the predicted label for x_t and $[\cdot]$ is the indicator function. The second one is mean absolute error (MAE) defined by $e = \frac{1}{|T|}\sum_{x_t \in T}|\hat{y}_t - y_t|$.

4.1 Results on the Historical Color Images Dataset

The historical color image dataset [Palermo *et al.*, 2012] is a benchmark dataset for algorithm evaluation, which includes

Table 1: Coding matrix of CNNOR

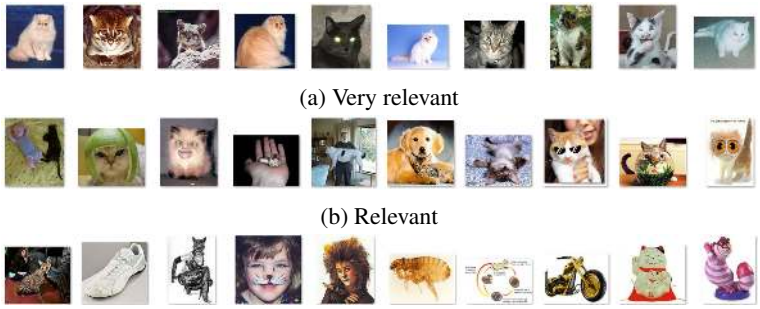
	$(x_t, 2)$	$(1, x_t, 3)$	$(2, x_t, 4)$	$(3, x_t, 5)$	$(4, x_t)$
1	+1	-1	-1	-1	-1
2	-1	+1	-1	-1	-1
3	-1	-1	+1	-1	-1
4	-1	-1	-1	+1	-1
5	-1	-1	-1	-1	+1



(a) 1930s (b) 1940s (c) 1950s

(d) 1960s (e) 1970s

Figure 3: Historical color image dating dataset.



(a) Very relevant

(b) Relevant

(c) Irrelevant

Figure 4: MSRA-MM1.0 dataset: cat subset.

Table 2: Baseline methods and experimental results.

Features	Methods		Accuracy (%)	MAE
Hand-crafted	Classification	Palermo et al.'s method [Palermo *et al.*, 2012]	44.92	0.93
	Ordinal regression	Martin et al.'s method [Martin *et al.*, 2014]	42.76	0.87
		RED-SVM [Lin and Li, 2012]	35.92	0.96
Deep learning	Classification	CNNm	41.07	1.06
	Ordinal regression	Niu et al.'s method [Niu *et al.*, 2016]	38.65	0.95
		CNNOR	41.56	1.04

Table 3: Accuracy performance of CNNm

#Iterations	2500	5000	7500	10000	12500	15000
Accuracy(%)	38.77	39.26	40.33	41.07	40.22	41.07

Table 4: Accuracy performance of Niu et al.'s method

#Iterations	7500	10000	12500	15000	17500	20000
Accuracy(%)	34.17	36.41	37.29	38.30	38.59	38.65

historical color images photographed in different decades. As shown in Figure 3, the dataset consists of five ordinal categories corresponding to five decades from 1930s to 1970s, and each category has 265 color images. For fair comparison, the same experimental setting as [Palermo *et al.*, 2012] is taken in this study. In each category, 215 images are selected for training and the rest 50 images are for testing. The same training and testing image partitions in [Palermo *et al.*, 2012] are used.

Two categories of baselines are used for comparison: hand-crafted feature based methods and deep learning methods. For each category, state-of-the-art multi-class classification methods and ordinal regression methods are evaluated as shown in Table 2. Palermo et al. [Palermo *et al.*, 2012] designed six categories of features for this task, which are 8168 dimensions in total. For fair comparison, all hand-crafted feature based methods in Table 2 are tested on these same features. Palermo et al. [Palermo *et al.*, 2012] focused on feature design for this specific task, and based on that they tackle the task as a multi-class classification problem by using a linear multi-class SVM as the classifier. Martin et al. [Martin *et al.*, 2014] improved the ordinal regression method for this particular task and got better MAE result. RED-SVM [Lin and Li, 2012] is a state-of-the-art general ordinal regression method but performs worse than Palermo et al. [Palermo *et al.*, 2012] and Martin et al. [Martin *et al.*, 2014] on this dataset.

The deep multi-class classification method (CNNm in Table 2) and the deep ordinal regression method (Niu et al.'s method in Table 2) based on CNN are implemented for comparison. The layers, parameters and organizations of the networks in CNNOR (i.e, G_1-G_5,G_i,G_j and G_k in Figure 2) are implemented exactly same as that in CNNm and Niu et al.'s method, except for the number of output neurons and the loss layer. Alex's architecture [Krizhevsky *et al.*, 2012] is employed in all the comparison methods. The image size of the historical dataset is equal or greater than 315×315 pixels, and we crop the images to 227×227 pixels for inputing to Alex's architecture. For each training/testing image partition, the last 5 images in the training set are used as the validation images, i.e., 210 images for training, 5 images for validation, and 50 images for testing in each rank. Thus, the total sizes of training, validation and testing sets for CNNm and Niu et al.'s method are 1050, 25 and 250 images. For CNN_{reg} of CNNOR, all the possible permutations of the images in the five ranks produce 210^5 training instances (i.e., the list $(x_i^1, x_i^2, \cdots, x_i^k, \cdots, x_i^m)$ in Eq. 1). In the experiments, 40960 instances are randomly selected from them to train CNN_{reg} and 40960 training triplets and 2100 validation triplets with equal numbers of positive and negative triplets are randomly selected to train CNN_k. In the testing phase, 30 triplets for each CNN_k are used to infer the label. The mini-

Table 5: Class distribution on MSRA-MM1.0 dataset

#Images	Rank 1	Rank 2	Rank 3	Total
Baby	379	295	277	951
Beach	336	398	213	947
Cat	243	344	378	965
Rose	222	418	329	969
Tiger	277	408	335	1020

Table 6: Accuracy (%) result on MSRA-MM1.0 dataset.

	Baby	Beach	Cat	Rose	Tiger
CNNm	48.00	50.67	47.56	55.11	53.33
Niu et al.	47.33	51.11	48.44	55.78	51.78
CNNOR	**51.56**	**56.45**	**52.67**	**59.78**	**60.00**

Table 7: MAE result on MSRA-MM1.0 dataset.

	Baby	Beach	Cat	Rose	Tiger
CNNm	0.667	0.598	0.676	0.522	0.571
Niu et al.	0.647	0.576	**0.620**	**0.500**	0.562
CNNOR	**0.640**	**0.551**	0.627	0.513	**0.523**

batch size is set to 64 in all the experiments. The learning rate for CNNm, Niu et al.'s method and CNN_{reg} are 0.01, and the learning rate for fine-tuning CNN_k from CNN_{reg} is 0.001. In the training phase, it is observed that the accuracies on validation sets of both CNNm and Niu et al.'s method are fluctuated dramatically because the validation sets are too small to reliably estimate the performance on the testing set. However, for CNNOR, the size of validation set is 33 mini-batches, and therefore, we can use the early stopping strategy to stop training when accuracy converges on the validation set. This is a benefit from the proposed method for small datasets for training and validation. In the experiments, the numbers of training iterations for CNNm and Niu et al.'s method need to be predefined. Table 3 and Table 4 show the test accuracies of CNNm and Niu et al.'s method on different numbers of iterations. After 7500 iterations for CNNm and 12500 iterations for Niu et al.'s method the losses on both training sets are smaller than 0.01. In Table 2, we choose the best accuracy from Table 3 and Table 4 for comparison. For CNNOR, the number of iterations to train CNNreg is 2500, and to train CNN_k is 2111, which is the average value on all 20 training/testing partitions. It shows that CNNOR outperforms the other two deep learning methods in terms of accuracy. Though Niu et al.'s method performs better than CNNOR in terms of MAE, its accuracy is significantly lower. It should be emphasized that the CNNm and Niu et al.'s results in Table 2 are the best results selected from Tables 3 and 4, where predefined iterations are used because the small validation sets cannot reliably estimate the testing accuracy. Comparing with the methods based on the hand-crafted features, CNNOR performs 5.64% better than RED-SVM in terms of accuracy and its performance is also comparable with the other two methods. Note that RED-SVM is an ordinal regression method for general ordinal regression problems but Palermo et al. and Martin et al.s methods, which are highly based on prior knowledge to design the classifiers and the features, are tailed-made for this dataset.

4.2 Results on the Image Retrieval Dataset

Microsoft Research Asia Multimedia 1.0 (MSRA-MM 1.0) dataset [Wang *et al.*, 2009] is a benchmark dataset to evaluate multimedia information retrieval algorithms and includes an image subset and a video subset. In the image dataset, 68 representative queries are selected based on the query log of Microsoft Live Search and then about 1000 images for each query are collected from the image search engine of Microsoft Live Search. The relevance annotations of the images are provided. For each image, its relevance to the corresponding query is labeled with three levels: very relevant, relevant and irrelevant. These three levels are indicated by rank 2, 1 and 0, respectively. Figure 4 shows an example of the "cat" query, where the first row lists images labeled as "very relevant", the second row shows some images labeled as "relevant", and the last row is "irrelevant" images. Given a testing image in a query set, predicting its relevance to the query is an ordinal regression problem. Hence, five subsets of MSRA-MM 1.0 image dataset which are "cat", "baby", "beach", "rose" and "tiger" are used to evaluate the performance of algorithms. The number of images in each rank of the five subsets is shown in Table 5. The total number of images in each subset is less than 1100 which is also a small dataset. The experiments on MSRA-MM 1.0 evaluate the CNNOR on data with different properties, including the different number of ranks, non-equal number of images in each rank and smaller image size.

The images in MSRA-MM 1.0 are thumbnails (i.e., the small images displayed on Microsoft Live Search) which are cropped to 3-channel 60×60 pixels in the experiments. The LeNet architecture [LeCun *et al.*, 1998] is employed in all the comparison methods: CNNm, Niu et al.'s method and CNNOR. The setting of mini-batch and learning rate are same as those in Section 4.1. Tables 6 and 7 summarize the results, which are the mean values on three random training/testing partitions. For accuracy, CNNOR performs 5.16% higher than CNN and 5.20% than Niu et al.'s methods averagely on the five subsets. For MAE, CNNOR achieves better results on three of the five subsets. Because there are not handcrafted features for MSRA-MM 1.0 dataset published in literature, to evaluate non-deep methods, the best baseline method in Table 2 using the 8168 features proposed for the historical dataset is tested. The accuracy on "cat" subset is 37.11%, which is 15.56% lower than CNNOR. The results indicate that CNNOR performs better than the two deep learning based multi-class classification method and ordinal regression method.

5 Conclusions

In this paper, a new ordinal regression algorithm is proposed for small data problems. CNNs are adapted to automatically extract high-level features to describe the ordinal relationship. To increase training data, this paper proposes a new network organization with triplets as instances and employs a new objective to pre-train the networks. Thus, deep learning can be applied more effectively on small dataset problems. The experimental results show that the proposed algorithm is comparable with the state-of-the-art methods.

Acknowledgments

This work was conducted within Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

References

[Cheng *et al.*, 2008] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1279–1284. IEEE, 2008.

[Chu and Ghahramani, 2005] Wei Chu and Zoubin Ghahramani. Gaussian processes for ordinal regression. In *Journal of Machine Learning Research*, pages 1019–1041, 2005.

[Chu and Keerthi, 2007] Wei Chu and S Sathiya Keerthi. Support vector ordinal regression. *Neural computation*, 19(3):792–815, 2007.

[Frank and Hall, 2001] Eibe Frank and Mark Hall. *A simple approach to ordinal classification*. Springer, 2001.

[Gutierrez *et al.*, 2016] Pedro Antonio Gutierrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *Knowledge and Data Engineering, IEEE Transactions on*, 28(1):127–146, 2016.

[Herbrich *et al.*, 1999] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, 1999.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lin and Li, 2012] Hsuan-Tien Lin and Ling Li. Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation*, 24(5):1329–1367, 2012.

[Liu *et al.*, 2011] Yang Liu, Yan Liu, and Keith CC Chan. Ordinal regression via manifold learning. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[Martin *et al.*, 2014] Paul Martin, Antoine Doucet, and Frédéric Jurie. Dating color images with ordinal classification. In *Proceedings of International Conference on Multimedia Retrieval*, page 447. ACM, 2014.

[McCullagh and Nelder, 1989] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

[Niu *et al.*, 2016] Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4920–4928, 2016.

[O'Connell, 2006] Ann A O'Connell. *Logistic regression models for ordinal response variables*. Number 146. Sage, 2006.

[Palermo *et al.*, 2012] Frank Palermo, James Hays, and Alexei A. Efros. Dating historical color images. In Andrew Fitzgibbon, Svetlana Lazebnik, Yoichi Sato, and Cordelia Schmid, editors, *ECCV (6)*, volume 7577 of *Lecture Notes in Computer Science*, pages 499–512. Springer, 2012.

[Shashua and Levin, 2002] Amnon Shashua and Anat Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. *Advances in neural information processing systems*, 15:937–944, 2002.

[Tian and Chen, 2015] Qing Tian and Songcan Chen. A novel ordinal learning strategy: Ordinal nearest-centroid projection. *Knowledge-Based Systems*, 88:144–153, 2015.

[Wang *et al.*, 2009] Meng Wang, Linjun Yang, and Xian-Sheng Hua. Msra-mm: Bridging research and industrial societies for multimedia information retrieval. *Microsoft Research Asia, Tech. Rep*, 2009.