A Deep Neural Network for Pairwise Classification: Enabling Feature Conjunctions and Ensuring Symmetry

Kyohei Atarashi[1], Satoshi Oyama[1], Masahito Kurihara[1], and Kazune Furudo[2]

[1] Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Hokkaido, Japan
{atarashi_k, oyama, kurihara}@complex.ist.hokudai.ac.jp,
[2] NEC Solution Innovators, Ltd., Tokyo, Japan
f.kazune08031991@gmail.com

Abstract. Pairwise classification is a computational problem to determine whether a given ordered pair of objects satisfies a binary relation R which is specified implicitly by a set of training data used for 'learning' R. It is an important component for entity resolution, network link prediction, protein-protein interaction prediction, and so on. Although deep neural networks (DNNs) outperform other methods in many tasks and have thus attracted the attention of machine learning researchers, there have been few studies of applying a DNN to pairwise classification. Important properties of pairwise classification include using feature conjunctions across examples. Also, it is known that making the classifier invariant to the data order is an important property in applications with a symmetric relation R, including those applications mentioned above. We first show that a simple DNN with fully connected layers cannot satisfy these properties and then present a pairwise DNN satisfying these properties. As an example of pairwise classification, we use the author matching problem, which is the problem of determining whether two author names in different bibliographic data sources refer to the same person. We show that the method using our model outperforms methods using a support vector machine and simple DNNs.

1 Introduction

Pairwise classification is a computational problem to determine whether a given ordered pair of objects satisfies a binary relation R which is specified implicitly by a set of training data used for 'learning' R. It is an important component for entity resolution, network link prediction, and so on. The method commonly used for identifying two objects includes defining a manually tuned similarity. However, defining suitable similarities is difficult. Therefore, machine learning techniques for learning from labeled data have been used.

When typical machine learning methods are applied to pairwise classification using the "Learn a classifier directly" approach, the design of the feature vector representing the pair of two objects is essential and important. Bilenko

and Moony [3] represented a pair of objects by using a feature vector based on common features between the two objects and a support vector machine (SVM). This method is effective for a problem like citation matching, where two objects belonging to the same class have many common features. However, if the two objects have few common features, this method is not effective. An example problem is the "author matching problem" in which the task is to determine whether two author names in different bibliographic data sources refer to the same person. Oyama and Manning [14] proposed applying a kernel method to this problem that uses the conjunctions of not only the common features but also those of different features across the two objects and using an SVM. Their method outperforms Bilenko and Moony's method in the author matching problem.

Deep neural networks (DNNs) have begun attracting attention in the field of machine learning as they have better performance than existing methods (e.g., SVM) in image classification [13], speech recognition [10], and many other tasks. While there have been many studies of applying a DNN to datum-wise classification, there have been few studies of applying a DNN to pairwise classification. Tran, Huynh, and Do [17] used a DNN as a classifier for the author matching problem. Since they represented feature vectors representing pairs of objects by concatenating fixed similarities and distance metrics, their approach does not take advantage of a DNN's ability to obtain feature representations automatically. A method using a DNN should be able to outperform existing methods even in pairwise classification by using feature vectors that enable a DNN to effectively learn feature representations.

A straightforward way of creating a feature vector representing a pair of objects is to concatenate the feature vectors of the two objects. The resulting vector can be used as input to a simple DNN with fully connected layers. The fully connected layers should enable even a simple DNN to use feature conjunctions across the two objects. In many applications such as entity resolution, the classifier results for training and prediction should be invariant with respect to the order of the pair values (the symmetry property).

In this paper, we first show that determining whether two objects satisfy relation R while satisfying symmetry is difficult for a simple DNN with fully connected layers. Next, we present a DNN-based model, i.e., a pairwise deep neural network (pairwise DNN), that ensures symmetry and enables feature conjunctions across examples. Then we present experimental results demonstrating that the method using our model outperforms other methods in the author matching problem.

2 Pairwise Classification

2.1 Problem Formulation

Pairwise classification is the computational problem to determine whether a pair of objects, $\boldsymbol{x}^{\alpha}$ and $\boldsymbol{x}^{\beta}$, satisfies relation R. Therefore, our goal is to obtain

classifier f:

$$f(\boldsymbol{x}^\alpha, \boldsymbol{x}^\beta) = \begin{cases} 1 \text{ (if } \boldsymbol{x}^\alpha \text{and } \boldsymbol{x}^\beta \text{satisfy } R) \\ -1 \text{ (otherwise).} \end{cases}$$

It is difficult to obtain accurate classifiers by using manually tuned similarities and thresholds. Therefore, machine learning methods in which learning is done from labeled data have been used. Many methods sample pair objects from the data, and then a person labels them in accordance with whether they satisfy relation R or not. Then these training examples are fed into classifier learning algorithms.

2.2 Symmetry

In many applications such as entity resolution, the classifier output should be invariant with respect to the order of the pair. Therefore, the classifier should satisfy symmetry; that is, $f(\boldsymbol{x}^\alpha, \boldsymbol{x}^\beta) = f(\boldsymbol{x}^\beta, \boldsymbol{x}^\alpha)$. Furthermore, the learning result should be invariant with respect to the order of the training data pairs.

3 Related Work

Machine learning approaches for determining the identity of two objects can be roughly grouped into three categories:

1. Learning a classifier directly.
2. Learning a similarity between two objects and deciding they are the same if the similarity exceeds a threshold.
3. Learning a distance between two objects and deciding they are the same if the distance is less than a threshold.

Then, toward lower approach, problem is more general and difficult. If the distance is obtained, the similarity can be obtained by reversing its sign. To learn the distance, it is necessary to satisfy the distance axiom. In many applications, there are cases in which only a classifier or a similarity suffice. In such cases, learning a classifier or a similarity is suitable according to Vapnik's principle [18]: "When solving a given problem, try to avoid solving a more general problem as an intermediate step". Furthermore, when learning a classifier directly, it is not necessary to set a threshold manually.

Bilenko and Moony [3] proposed a method for learning a classifier directly for the citation matching problem, in which a determination is made as to whether two citations in different bibliographic data sources refer to the same paper. They represent the original object as a bag-of-words feature vector $\boldsymbol{x}^\alpha = (x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha)^{\mathrm{T}}$ and a pair object consisting of $\boldsymbol{x}^\alpha = (x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha)^{\mathrm{T}}$ and $\boldsymbol{x}^\beta = (x_1^\beta, x_2^\beta, \ldots, x_n^\beta)^{\mathrm{T}}$ as feature vector $\boldsymbol{x}_{\mathrm{Hadamard}}$:

$$\hat{\boldsymbol{x}}_{\mathrm{Hadamard}} = (x_1^\alpha x_1^\beta, x_2^\alpha x_2^\beta, \ldots, x_n^\alpha x_n^\beta)^{\mathrm{T}}. \tag{1}$$

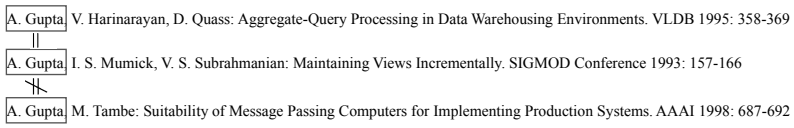

Fig. 1. Matching authors

This feature vector is a Hadamard product of $\boldsymbol{x}^\alpha$ and $\boldsymbol{x}^\beta$. They labeled it y and used an SVM. The classifier satisfies the symmetry because the feature vectors representing the pair objects satisfy symmetry.

Bilenko and Mooney's method is effective for a problem like citation matching, in which two objects from the same class have many common features, but it is not effective if the two objects from the same class have few common features because it cannot distinguish between positive pairs and negative pairs. An example of this is the author matching problem, in which a determination is made as to whether two author names in different bibliographic data sources refer to the same person. In the illustrative example in Figure 1, where A. Gupta is the abbreviated form of Ashish Gupta and Anoop Gupta, the first two records have no common words even though A. Gupta is the same person in both cases. One approach to such problems includes using not only common features but also conjunctions of different features across examples. Oyama and Manning [14] proposed using $\hat{\boldsymbol{x}}_{\mathrm{Cartesian}}$,

$$\hat{\boldsymbol{x}}_{\mathrm{Cartesian}} = (x_1^\alpha x_1^\beta, \ldots, x_1^\alpha x_n^\beta, x_2^\alpha x_1^\beta, \ldots, x_2^\alpha x_n^\beta, \ldots, x_n^\alpha x_1^\beta, \ldots, x_n^\alpha x_n^\beta)^{\mathrm{T}}. \quad (2)$$

as a feature vector representing a pair of objects. This is the Cartesian product between $\boldsymbol{x}^\alpha$ and $\boldsymbol{x}^\beta$. However, the dimension of this feature vector is n^2, so doing this straightforwardly is computationally intensive. They thus represented a pair object by using feature vector $\hat{\boldsymbol{x}}_{\mathrm{concat}}$,

$$\hat{\boldsymbol{x}}_{\mathrm{concat}} = \begin{pmatrix} \boldsymbol{x}^\alpha \\ \boldsymbol{x}^\beta \end{pmatrix} = (x_1^\alpha, \ldots, x_n^\alpha, x_1^\beta, \ldots x_n^\beta)^{\mathrm{T}}, \quad (3)$$

which concatenates original objects, and proposed using the following kernel:

$$K(\hat{\boldsymbol{x}}_{\mathrm{concat}}, \hat{\boldsymbol{z}}_{\mathrm{concat}}) = \langle \hat{\boldsymbol{x}}_{\mathrm{Cartesian}},\ \hat{\boldsymbol{z}}_{\mathrm{Cartesian}} \rangle. \quad (4)$$

Using the SVM with this kernel enables classification on the Cartesian product space between two objects without high computational cost. Although this feature vector is not symmetrical, they showed that including a pair that is reversed with respect to the concatenation order or symmetrizing the kernel ensures classifier symmetry. A similar method has been used to predict protein-protein interactions [1].

As mentioned in the Introduction, DNNs are attracting attention in the field of machine learning as methods using them have better performance than existing methods in many tasks. Since DNNs can automatically obtain feature

representations, a method using them should be able to outperform existing methods even in pairwise classification.

Also as mentioned in the Introduction, while there have been many studies of applying a DNN to datum-wise classification, there have been few studies of applying a DNN to pairwise classification, and most of these studies used learning of similarity or distance metric. Bromley et al. [5] proposed using a Siamese network for handwritten signature pair determination. In a Siamese network, two objects are input to a NN individually, and the similarity or distance between the outputs of the NN is output. Also, there have been several proposed methods based on a Siamese network for face verification [6] [16] [11]. Tran, Huynh, and Do [17] applied a DNN as a classifier to the author matching problem. Since they represented feature vectors representing pairs of objects by concatenating fixed similarities or distance metrics, this approach cannot utilize a DNN's ability to obtain representations automatically. A method using a DNN should be able to outperform existing methods even in pairwise classification by using feature vectors to effectively learn feature representations.

4 Problem of DNN in Pairwise Classification

Because only one object can be input to a DNN, it is necessary to design a feature vector representing a pair of objects. Given that a DNN can automatically obtain feature representations, the feature vector should self-sufficiently represent the information of the two objects. Hence, we first present a straightforward design of a feature vector representing a pair of objects that concatenates the feature vectors of the two objects, as given by Equation (3). Because the input and hidden layers of a simple DNN are fully connected, the DNN should provide feature representation considering feature conjunctions across examples.

Since the feature vector represented by Equation (3) does not satisfy symmetry, a symmetric DNN should be used when this feature vector is input. Bishop [4] classified the approaches to making a NN invariant into four approaches

1. The training set is augmented using replicas of the training patterns, transformed in accordance with the desired invariance.
2. A regularization term is added to the error function that penalizes changes in the model output when the input is transformed.
3. Invariance is built into the pre-processing by extracting features that are invariant under the required transformations.
4. Invariance is built into the structure of the NN.

In pairwise classification, the NN should be invariant with respect to the order of the data values in a pair; i.e., it should have symmetry. Approach 1, called data augmentation, incurs extra computational costs. Approach 2 cannot be used in pairwise classification because transformation that changes the data order is not continuous. Approach 3 is problematic because designing suitable features is difficult. Therefore, we took Approach 4. If the values of the hidden layer connected with the input layer satisfy symmetry, the NN output satisfies

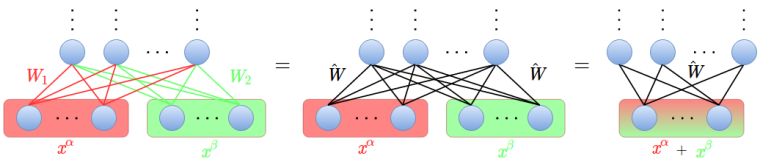

Fig. 2. Symmetric DNN

symmetry. Namely, let $\boldsymbol{W}$ be the weight matrix between the input and the hidden layer. If the following equation holds, the NN output satisfies symmetry.

$$\boldsymbol{W}\begin{pmatrix}\boldsymbol{x}^{\alpha}\\ \boldsymbol{x}^{\beta}\end{pmatrix} = \boldsymbol{W}\begin{pmatrix}\boldsymbol{x}^{\beta}\\ \boldsymbol{x}^{\alpha}\end{pmatrix} \tag{5}$$

Let $\boldsymbol{W}_1$ be the weight matrix between the partial input layer for the first object and the entire hidden layer, $\boldsymbol{W}_2$ be the weight matrix between remaining input layer for the second object and the entire hidden layer. Since $\boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2)$, Equation (5) can be rewritten: $\boldsymbol{W}_1(\boldsymbol{x}^{\alpha} - \boldsymbol{x}^{\beta}) = \boldsymbol{W}_2(\boldsymbol{x}^{\alpha} - \boldsymbol{x}^{\beta})$. For arbitrary $\boldsymbol{x}^{\alpha}$ and $\boldsymbol{x}^{\beta}$, this equation holds if $\boldsymbol{W}_1 = \boldsymbol{W}_2$. However, there is actually a problem here. If this equation holds, the following equation holds.

$$\boldsymbol{W}\boldsymbol{x}_{\text{concat}} = \hat{\boldsymbol{W}}(\boldsymbol{x}^{\alpha} + \boldsymbol{x}^{\beta}), \tag{6}$$

where $\hat{\boldsymbol{W}}$ is equal to $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. From Equation (6), it follows that the addition of feature vectors of two objects is input to the DNN. However, the addition of feature vectors of two objects is not suitable for pairwise classification because the information about the features of the individual objects is lost. For example, in the author matching problem, it cannot be determined which words appear in which papers.

5 Proposed Method

5.1 Pairwise DNN

Our proposed pairwise DNN ensures symmetry without losing information about the features of individual objects and enables feature conjunctions across examples. In a pairwise DNN, two objects, $\boldsymbol{x}^{\alpha}$ and $\boldsymbol{x}^{\beta}$, are first mapped individually to $\boldsymbol{z}^{\alpha} = \hat{\boldsymbol{W}}\boldsymbol{x}^{\alpha}$ and $\boldsymbol{z}^{\beta} = \hat{\boldsymbol{W}}\boldsymbol{x}^{\beta}$, where $\hat{\boldsymbol{W}}$ is a common $m \times n$ weight matrix. Next, a Hadamard layer, which calculates the Hadamard product of $\boldsymbol{z}^{\alpha}$ and $\boldsymbol{z}^{\beta}$, is introduced, and $\hat{\boldsymbol{z}} = (z_1^{\alpha} z_1^{\beta}, z_2^{\alpha} z_2^{\beta}, \ldots, z_m^{\alpha} z_m^{\beta})^{\mathrm{T}}$ is calculated. Finally, $\hat{\boldsymbol{z}}$ is input to a simple fully connected DNN. $\hat{\boldsymbol{W}}$ is not a fixed parameter but a parameter that is learned when a simple fully connected DNN is trained by stochastic gradient descent (SGD) with backpropagation. Figure 3 provides an overview of a pairwise DNN.

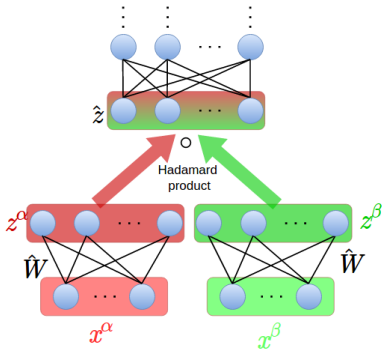

Fig. 3. Pairwise DNN

5.2 Symmetry

Since a Hadamard product is commutative and $\hat{\boldsymbol{W}}$ is common between two objects, the output of a pairwise DNN satisfies symmetry. Let $\hat{\boldsymbol{w}}_i$ be the ith row vector of $\hat{\boldsymbol{W}}$ and w_{ij} be the jth element of $\hat{\boldsymbol{w}}_i$. Then $\hat{z}_i$, the ith element of $\hat{\boldsymbol{z}}$, can be written as

$$\hat{z}_i = \sum_{k=1}^{n} \sum_{l=1}^{n} \hat{w}_{ik} \hat{w}_{il} x_k^{\alpha} x_l^{\beta}. \tag{7}$$

In the elements of $\hat{\boldsymbol{z}}$, only $\hat{z}_i$ depends on $\hat{w}_{ij}$. The partial differential of $\hat{z}_i$ with respect to w_{ij} is given by

$$\frac{\partial \hat{z}_i}{\partial \hat{w}_{ij}} = \sum_{k=1}^{n} (\hat{w}_{ik} (x_j^{\alpha} x_k^{\beta} + x_k^{\alpha} x_j^{\beta})). \tag{8}$$

Since this satisfies symmetry, the learning result satisfies symmetry. Therefore, a pairwise DNN satisfies symmetry.

A pairwise DNN can be considered to extract a feature $\hat{\boldsymbol{z}}$. However, since this feature extraction itself is also learned, invariance is built into the structure of a pairwise DNN. This is Approach 4 described above, making the NN invariant as classified by Bishop [4].

5.3 Feature Conjunctions across Examples

In Equation (7), which formulates $\hat{z}_i$, $x_k^{\alpha} x_l^{\beta}$ is the feature conjunction of the kth feature of the first object and lth is the feature of the second object. Therefore, each element of $\hat{z}_i$ can be considered to be the weighted summation of feature conjunctions across examples. Our approach is the same as Bilenko and Moony's with respect to using a vector expressed by a Hadamard product but different with respect to enabling feature conjunctions across examples by using the Hadamard product of a mapped vector. Like Oyama and Manning's approach, our approach using feature conjunctions should be effective when two objects belonging to the same class have few common features.

Even though the mapping of each object is linear, like a general NN, it is possible to use a non-linear activation function. However, if non-linear activation function $h(\cdot)$ is used, the ith element of $\hat{\boldsymbol{z}}$ becomes $\hat{z}_i = h(z_i^\alpha)h(z_i^\beta)$, so a pairwise DNN does not always enable feature conjunctions across examples. For example, let $h(\cdot)$ be a sigmoid function. Then $\hat{z}_i$ becomes:

$$\hat{z}_i = \frac{1}{1 + \exp(-z_i^\alpha) + \exp(-z_i^\beta) + \exp(-z_i^\alpha - z_i^\beta)}.$$

Since $z_i^\alpha = \sum_{j=1}^n \hat{w}_{ij} x_j^\alpha$ and $z_i^\beta = \sum_{j=1}^n \hat{w}_{ij} x_j^\beta$, feature conjunctions do not appear in $\hat{z}_i$.

6 Experiment

6.1 Dataset and Overview

We experimentally evaluated our proposed method using the author matching problem on the DBLP dataset, which is a bibliography of computer science papers. We extracted 3,384 papers for which there were 729 unique author names. We only used papers for which the full name of author was given. Papers with the same author name were assumed to have been written by the same person. For the author matching problem, we abbreviated first names into initials and removed middle names. We used all words appearing in their titles, coauthor names and publication venues. Each original object was represented by a bag-of-words feature vector and the dimension of all vectors was 9,264.

There are two methods for creating a training set and a test set: (1) First, create a pair set from the original objects set. Next, split the pair set into a training set and a test set. (2) First, split the original objects set into a training objects set and a test objects set. Next, create a training set and a test set by making pairs from each objects set. With the method (1), the problem is easy because the objects constituting the test set are also in the training set. We thus used the method (2). If the authors specified by the objects of the pair are the same person, the pair was given a positive label, and if authors are not the same person, the pair was given a negative label. We only used pairs constituted by two papers with the same abbreviated author name.

Table 1. Size of dataset

	Fold 1	Fold 2
No. of papers	1,692	1,692
No. of pair objects	22,264	22,416

If the training set and test set are made using method (2), the number of negative pairs is much larger than that of positive pairs. We thus balanced the two sets by sampling the negative pairs, as described elsewhere [3]. The sizes of

the datasets are shown in Table 1. Because author matching problem is binary classification problem, we evaluated accuracy, precision, and recall on the test set. Since which precision or recall is more important depends on situation, we evaluated the precision-recall curve and the area under the curve (AUC), as discussed elsewhere [2].

6.2 Experiment 1: Comparison of Proposed Method with Other Methods

In Experiment 1, we compared five methods:

Pairwise DNN Proposed method. The number of units in each layer was 1000, and the number of hidden layers behind the Hadamard layer was 3. The mapping of each objects was linear.

Concat DNN A simple fully connected DNN with feature vectors of pairs of objects represented by Equation (3). The number of units in each layer was 1000, and the number of hidden layers was 4.

Concat DNN(aug) Same as Concat DNN except that feature vectors with the concatenation order reversed were included in the training set. Also, at the time of prediction, feature vectors with the concatenation order reversed were predicted, and the mean of the two outputs was used as the final output. In short, data augmentation was performed on the training and test sets.

Addition DNN A simple fully connected DNN with feature vectors of pairs of objects represented by the addition of two objects. This model is equivalent to a symmetric DNN, as illustrated in Figure 2. The number of units in each layer was 1000, and the number of hidden layers was 4.

Pairwise SVM An SVM with the kernel proposed by Oyama and Manning [14].

In the DNN-based methods, the ReLU activation function [9] was used for the hidden layers, a sigmoid function was used for the output layer, and a loss function was used for cross entropy. Dropout [15] was used for all hidden layers and the Hadamard layer. All weight were initialized using Glorot's method [8], which uses a uniform distribution with the interval adjusted in accordance with the number of units.

We used SGD with a mini-batch of size 128 for 100 epochs. The Adam optimizer [12] was effective for **Pairwise DNN**. The AdaGrad optimizer [7] was effective for **Concat DNN**, **Concat DNN(aug)**, and **Addition DNN**. The default learning rate of Adam is 0.001, but we found that a learning rate several times higher produced better results. We thus set the learning rate to 0.002. We set it to 0.005 for the methods using AdaGrad. For hyper-parameter C of **Pairwise SVM**, we tested using 21 values, $[2^{-10}, 2^{-9}, \ldots, 2^{10}]$, and found that ten test values produced comparably good results. We thus set $C = 2^0 = 1$.

As shown in Table 2, **Pairwise DNN** had the highest recall, accuracy, and AUC. Although there is a trade-off relationship between precision and recall, the precision of **Pairwise DNN** was kept high. As shown in Figure 4, **Pairwise DNN** and **Pairwise SVM**, both of which enable feature conjunctions across

Table 2. Precision (P), recall (R), AUC, and accuracy (Acc) for five methods.

Method	P	R	AUC	Acc
Pairwise DNN	0.938	**0.717**	**0.924**	**0.835**
Concat DNN	0.881	0.600	0.869	0.760
Concat DNN(aug)	0.908	0.564	0.869	0.754
Addition DNN	0.932	0.413	0.839	0.692
Pairwise SVM	**0.968**	0.540	0.918	0.761

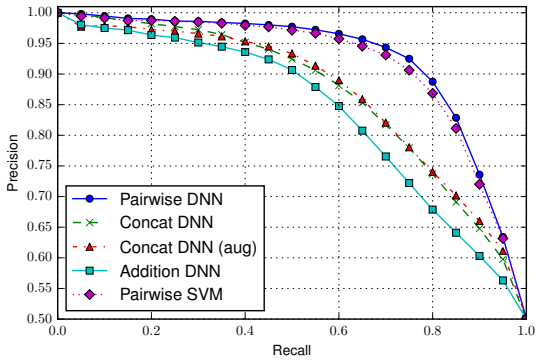

Fig. 4. Precision-recall curves for five methods.

examples, retained high precision at higher recall levels, especially **Pairwise DNN**. The results of **Concat DNN(aug)** and **Concat DNN** indicated that data augmentation was ineffective in this experiment.

6.3 Experiment 2: Evaluation of Proposed Method

In Experiment 2, we evaluated the proposed pairwise DNN. First, we changed the number of hidden layers behind the Hadamard layer, $[0, 1, 2, 3, 4, 5, 6]$. The results for **Pairwise DNN** in Experiment 1 (in which there were three hidden layers) were used as a baseline. The number of parameters in each model was made almost the same by adjusting the number of units. The other conditions were the same as for Experiment 1. As shown in Table 3, as the number of hidden layers was increased, recall and accuracy tended to increase while precision tended to decrease.

Next, we compared four models whose mapping of each object was linear (**Linear**) and non-linear (**Sigmoid**, **Tanh**, and **ReLU**). The other conditions (e.g., the number of hidden layers) were the same as for **Pairwise DNN** in Experiment 1. As shown in Table 4, the results with **Sigmoid** and **ReLU** were significantly worse than with **Linear** except for precision. Clearly, a pairwise DNN does not enable feature conjunctions across examples when mapping each object with **Sigmoid** or **ReLU**. In contrast, the results with **Tanh** were nearly

Table 3. Precision (P), recall (R), AUC, and accuracy (Acc) for seven models with different numbers of hidden layers (n = number of hidden layers).

Models	P	R	AUC	Acc
Layer 0	0.966	0.599	0.905	0.789
Layer 1	**0.971**	0.624	**0.926**	0.803
Layer 2	0.936	0.693	0.921	0.823
Layer 3	0.938	0.717	0.924	0.835
Layer 4	0.922	0.755	0.925	0.845
Layer 5	0.921	0.734	0.905	0.836
Layer 6	0.920	**0.757**	0.925	**0.846**

Table 4. Precision (P), recall (R), AUC, and accuracy (Acc) for four models whose mapping of each object was linear and non-linear.

Models	P	R	AUC	Acc
Linear	0.938	**0.717**	0.924	**0.835**
Sigmoid	0.913	0.453	0.799	0.705
Tanh	0.949	0.706	**0.925**	0.834
ReLU	**0.964**	0.569	0.898	0.774

the same as those with **Linear**. Clearly, **Tanh** is almost linear for low absolute input values, and the weights are initialized using a uniform distribution with a mean of 0.

7 Conclusion and Future Work

Pairwise classification is an important component in entity resolution, network link prediction, protein-protein interaction prediction, and so on. We showed that there is a problem regarding symmetry and object information when a simple DNN is learned directly as a classifier with feature vectors represented by concatenating two objects. Our proposed DNN-based model, a pairwise DNN, satisfies symmetry without losing the information of each object and enables feature conjunctions across examples. Experimental results for the author matching problem showed that a pairwise DNN had higher recall, accuracy, and AUC. Experiments on a pairwise DNN clarified several properties.

This work focused on the author matching problem. Future work includes application of the pairwise DNN to other problems (e.g., face verification) and extension of the pairwise DNN model to convolutional neural networks, recursive neural networks, and so on.

References

1. Ben-Hur, A., Noble, W.S.: Kernel methods for predicting protein–protein interactions. Bioinformatics 21(suppl 1), i38–i46 (2005)

2. Bilenko, M., Mooney, R.: On evaluation and training-set construction for duplicate detection. In: Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation. pp. 7–12 (2003)
3. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 39–48 (2003)
4. Bishop, C.: Patern Recognition and Machine Learning, pp. 261–267. Springer (2006)
5. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: Advances in Neural Information Processing Systems. pp. 737–744 (1993)
6. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. vol. 1, pp. 539–546 (2005)
7. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12, 2121–2159 (2011)
8. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics 9, 249–256 (2010)
9. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. vol. 15, p. 275 (2011)
10. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29(6), 82–97 (2012)
11. Hu, J., Lu, J., Tan, Y.P.: Discriminative deep metric learning for face verification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1875–1882 (2014)
12. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
14. Oyama, S., Manning, C.D.: Using feature conjunctions across examples for learning pairwise classifiers. In: Proceedings of the 15th European Conference on Machine Learning. pp. 322–333 (2004)
15. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958 (2014)
16. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1701–1708 (2014)
17. Tran, H.N., Huynh, T., Do, T.: Author name disambiguation by using deep neural network. In: Asian Conference on Intelligent Information and Database Systems. pp. 123–132. Springer (2014)
18. Vapnik, V.: The Nature of Statistical Learning Theory, p. 30. Springer (2000)