Preference Neural Network

Ayman Elgharabawy, Mukesh Prasad Senior Member, *IEEE*, Nikhil R. Pal, Fellow *IEEE*, Chin-Teng Lin, Fellow *IEEE*

***Abstract*—Equality and incomparability multi-label ranking have not been introduced to learning before. This paper proposes new native ranker neural network to address the problem of multi-label ranking including incomparable preference orders using a new activation and error functions and new architecture. Preference Neural Network *PNN* solves the multi-label ranking problem, where labels may have indifference preference orders or subgroups which are equally ranked. *PNN* is a non-deep, multiple-value neuron, single middle layer and one or more output layers network. PNN uses a novel positive smooth staircase (*PSS*) or smooth staircase (*SS*) activation function and represents preference orders and Spearman ranking correlation as objective functions. It is introduced in two types, Type A is traditional *NN* architecture and Type B uses expanding architecture by introducing new type of hidden neuron has multiple activation function in middle layer and duplicated output layers to reinforce the ranking by increasing the number of weights. *PNN* accepts single data instance as inputs and output neurons represent the number of labels and output value represents the preference value. *PNN* is evaluated using a new preference mining data set that contains repeated label values which have not experimented on before. *SS* and PS speed-up the learning and *PNN* outperforms five previously proposed methods for strict label ranking in terms of accurate results with high computational efficiency.**

***Index Terms*—Preference learning, Multi-label ranking, Neural network, Kendall's tau, Preference mining**

I. Introduction

PREFERENCE learning (*PL*) is emerging as an extended paradigm in machine learning by inducing predictive preference models from experimental data [?] [?] [?]. *PL* has applications in a variety of research areas such as knowledge discovery and recommender systems [?]. Objects, instances and label ranking are the three main categories of *PL*. Of those, label ranking (*LR*) is a challenging problem that has gained importance in information retrieval by search engine [?] [?]. Unlike the standard problems of regression and classification, multi-label ranking involves predicting the relation between the orders of multiple labels. In case of multiclass classification, for a given instance x from the instance space $\mathbb{X}$, there is a label y associated with x, where $y \in \mathscr{L}$, where $\mathscr{L} = \{\lambda_1,..,\lambda_n\}$, where n is the number of labels. *LR* is an extension of multi-class and multi-label classification, where each instance x is assigned an ordering of all the class labels in the set $\mathscr{L}$. This ordering gives the ranking of the labels for the given object x. This ordering can be represented by a permutation of the set $\{1,2,\cdots,n\}$. The label order has the following three features. Irreflexive where $\lambda_a \not\succ \lambda_a$,Transitive where $\lambda_a \succ \lambda_b \wedge \lambda_b \succ \lambda_c \implies \lambda_a \succ \lambda_c$ and Asymmetric $\lambda_a \succ \lambda_b$ then $\lambda_b \not\succ \lambda_a$. Label preference takes one of two forms, Strict and Non-Strict order. The strict label order $(\lambda_a \succ \lambda_b \succ \lambda_c \succ \lambda_d)$ can be represented as π=(1,2,3,4) and for non-restricted total order π=$(\lambda_a \succ \lambda_b \simeq \lambda_c \succ \lambda_d)$ can be represented as $\pi = (1,2,2,3)$, where a, b, c and d are the label indexes and λ_a, λ_b, λ_c and λ_d are the ranking values of these labels respectively.

For the non-continuous permutation space, The order can be represented by the aforementioned relations in addition to the incomparability binary relation $\perp$. For example the partial order $\lambda_a \succ \lambda_b \succ \lambda_d$ can be represented as π=(1,2,0,3) where 0 represents an incomparable relation since λ_c is not comparable to $(\lambda_a,\lambda_b,\lambda_d)$

Various label ranking methods have been introduced in recent years [?]such as Decomposition based methods, Statistical methods, Similarity and Ensemble based methods.

Decomposition methods include pairwise comparison [?] [?], log linear models and constraint classification [?]. The pairwise approach introduced by hüllermeier [?] divides the multi-label ranking problem into several binary classification problems in order to predict the pairs of labels $\lambda_i \succ \lambda_j$ or $\lambda_j \prec \lambda_i$ foran input x. Statistical methods includes decision trees [?], instance based methods (Plackett-Luce) [?] and Gaussian mixture model [?] based approaches.For example, [?] use Gaussian Mixture models to learn soft pairwise label preferences. Similarity methods minimize the labels distance instead of maximizing the probability of label values. For example, Aiguzhinov et al. [?] used an adapted version of Naive Bayes algorithm which used similarity between label rankings instead of probability. Association rules mining has also been adapted for label ranking [?]. The multilayer perceptron has also been adapted for label ranking [?], [?]. For example, in [?] six approaches are proposed that use label ranking loss during back propagation.

Instance-based decision tree was introduced by Cheng and Hüllermeier to rank the labels based on predictive probability models of a decision tree [?]. Hüllermeier combined both a decision tree and supervised clustering in two approaches for label ranking by mapping between instances and multi-labels ranking space [?].

Neural Networks (*NN*) for ranking were first introduced as (RankNet) by Burge to solve the problem of object ranking for sorting web documents by search engine [?]. Rank-net uses Gradient descent and probabilistic ranking cost function for each object pair. Multilayer perceptron (*MLP-LR*) for label ranking [?] employs *NN* architecture using sigmoid activation function to calculate the error between the actual and expected values of the output labels. However, It uses local approach it minimizes the individual error per each output neuron by subtract actual - predicted value and uses Kendall error as global approach. Both approaches doesn't use ranking objective function in BP and learning steps. Zhou and Yangming provided a scalable decision tree structure by implementing a random forest with a parallel computational architecture for extreme label ranking [?]. Claudio Rebelo introduced label random forest (*LRF*) as an ensemble method of ranking [?]. *LRF* was based on the best approach result of previous ranking decision trees using entropy-based ranking. Jung and Tewari proposed an approach for label ranking based on voting of the best learners and scoring the labels for ranking [?]. Song and Huang proposed a framework to solve the vulnerability of multi-label deep learning models [?]. Yan and Wang proposed a long short term memory (*LSTM*) based multi-label ranking model for document classification [?]. This method uses a two-step process - the first step learns the document representation while the second step ranks the labels. Guo and Hou introduced low rank multi-label classification with missing labels (*LRML*) which recovered the missing labels via Laplacian manifold regularization derived from the feature space by utilizing the low-rank mapping [?]. In our study we shall use stair-case activation functions. In this context it is worth noting that Staircase activation function has been used for rule extraction from a multi-layer perceptron network [?]. Moraga and Heider [?] proposed a Generalized Multi-pleâĂŞvalued Neuron with a differentiable soft staircase activation function which is represented by a sum of a set of sigmoidal functions.

Some of the above mentioned methods and their variants have some issues which can be broadly categorized into two types:

(i) Drawbacks of classification based models: When the ranking model is constructed using binary classification models of an associated higher dimensional label space, such a method cannot take into account interaction between labels. In this case such rankings based on minimizing pairwise classification error is not necessarily equivalent to maximizing the performance of the label ranking considering all labels.

(ii) Drawbacks of Probability based models: Some such methods use probabilistic scores for individual classes for ranking the labels. Such methods cannot capture dependence between labels and do not take into account distance between labels.

The proposed *PNN* enjoys several advantages over existing in multi-label ranking methods.

1) *PNN* uses gradient ascent to maximize spearman ranking correlation coefficient which helps to provide better order for each label for both learning and stopping condition. Whereas other methods such as *MLP-LR* uses individual output error by subtracting actual- predicted ranking which may not give best ranking results.
2) *PNN* is implemented directly as a ranker *NN*. It uses staircase activation functions to rank all the labels simultaneously. The *SS* or *PSS* functions provide multiple output values during the conversions; whereas the exiting *NN*-based methods based such as *MLP-LR* and *Rank-net* use Sigmoid and Relu activation functions. These activation functions has a binary output. Thus, it solves the ranking problem as pairwise classification.
3) *PNN* can rank the indifference preference order where the number of ranking values is less than the number of labels. To address this problem, *PNN* assigns the same ranking value to more than one output neuron. However, existing methods fail to address this issue.
4) *PNN* is applied the ranking problem related to incomparability preference order by assigning label 0 (zero) to the unrelated label. For example, the partial order $\lambda_1 \succ \lambda_2 \succ \lambda_4$ can be represented as π = (1, 2, 0, 3), where 0 represents incomparable relation $\perp$ of LR expression (λ_1, λ_2, λ_4) $\perp$ λ_3, Where existing methods do not address this problem due to lack of dataset.

II. Initial Ranking Neural Network

This proposed research is based on initial experiment to implement simple ranker neural network using Kendall tau error function and sigmoid activation function as mention in fig. **??**.

As shown from Fig. **??**, The learning convergence to rank 2 different multi labels order data set takes 160 epochs using sigmoid function, Thus generate equal numbers will take more iterations to rank equality because of sigmoid shape which has slightly high rate of change of y value for each x value. therefore we consider ranking performance is one the disadvantage of Sigmoid *PNN*. The video shows the actual learning conversion of *PNN* Sigmoid function in the following link [?].

III. *PNN* architecture

A. Activation Functions

Existing *NN* activation functions are monotonic and it delivers binary output or range of values. they produce binary output based on threshold value which determine the final output. Non of the existing functions produce multiple fixed integer values along x axis similar to step function because absolute step are not differentiable. We propose two non-linear activation functions have differentiable staircase shape. these functions similar to the staircase mention in [?], however we use tanh as a polynomial instead of exponential due to implementation problem. This *SS* shape detect consecutive integer values and transition from low to high rank (or vice versa) is fast and not interfered with threshold detection. staircase is similar to step function which we can call it diffrentiable smooth step function. The width of staircase keep the stability of the ranking during the forward and backward propagation and make the convergence is smooth.

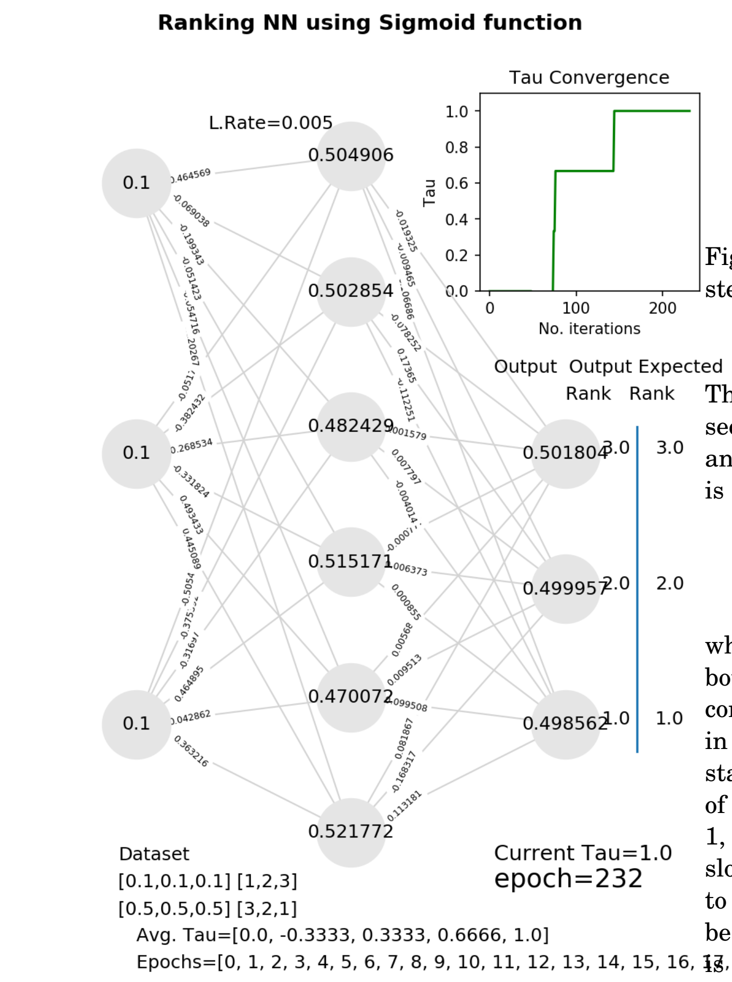


Fig. 1: Sample output image of video file demonstrates the evaluation of *NN* using Sigmoid Activation for ranking. [?]

1) Positive Smooth Staircase (PSS): PSS is a non-linear and monotonic sigmoid type activation functions represented as a bounded smooth staircase function starts from x=0 to ∞. This proposed function is called positive smooth staircase (*PSS*), as shown in Fig. 1. *PSS* is a polynomial of multiple $tanh(x)$ functions and is therefore differentiable and continuous. The function squash the output neurons during the forward propagation into finite multiple integer values. These values represents the preference values from {*0* to n} where 0 represents the incomparable relation $\perp$ and values from 1 to n represent the label ranking. The input features should be normalized from 0 to 1. The activation function is given in Eq. **??**.

$$y = \frac{1}{2} \cdot \Big(\sum_{i=0,1,..}^{n} \tanh(w \cdot x - (s \cdot i)) + n \Big) \tag{1}$$

Where n is number of output labels,w i s the step factor = 10, s is the step smoothness s value start from 5 to 130.

2) Smooth Staircase (SS): The proposed (*SS*) represents staircase similar to (*PSS*). However, *SS* has fixed sharp edge and it has variable boundary value to cover a wide range of input values of normalized features from -1 to 1.

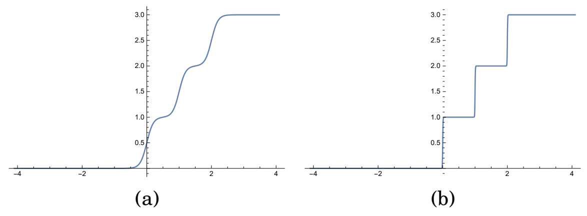


Fig. 2: *PSS* activation function diagram where n=3 ,where step smoothness s=5 and 100 in (a) and (b) respectively.

The derivative of the activation function is discussed in section III and the performance comparison between *SS* and *PSS* are mention in section v. The activation function is given in Eq. **??**.

$$y = -\frac{1}{2} \cdot \Big(\sum_{i=0,1,..}^{n} \tanh(\frac{-100}{b} \cdot x + c(1 - \frac{2i}{n-1})) + \frac{n}{2} \Big) \tag{2}$$

where c=100, n = number of ranked labels and b is the boundary value. where (*SS*) lies between -b and b. c is constant are chosen to formulate the shape of SS function in x and y axis. The (*SS*) function has the shape of smooth stair steps , where each step represents integer number of label ranking in y-axis from *0* to ∞. as shown in Fig. 1, The *SS* step is not absolute flat but it has differential slope, The function boundary on x-axis has range from -∞ to ∞ Therefore, *NN* input feature values are required to be normalized from -1 to 1 on the x-axis. The step width is 1 when n≃2b. The convergence rate based on the step width. However it may take less time to converge based on *NN* hyper parameters. The Fig. **??** (a) and (b) shows the activation functions to rank 6 and 9 labels respectively.

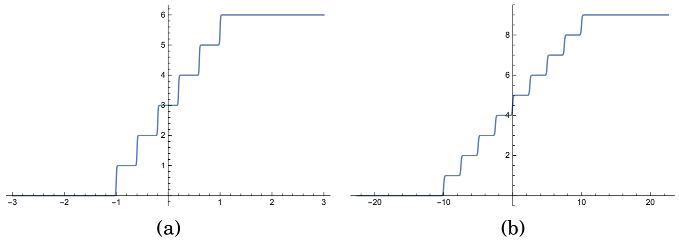


Fig. 3: *SS* activation function where n=6 and boundary b=1 in (a). n=9 and boundary b=10 in (b)

B. Ranking high number of labels

By increasing number of labels, Data separability decreases. We use Fishe LDA to measure the separability of the data. Scaling value for features and activation function is calculated using Eq. **??**

$$S = 100 \cdot (1 - \lambda_i) \tag{3}$$

where λ_i is the largest eigen value. The scaling up technique is applied in 2 steps.

(i) Increase SS activation function boundaries by increasing b value of equation to make the step width not less than 1.

(ii) Increase the data separability of features by scaling the data up to the SS activation function boundaries. as shown in Fig **??**. i.e, S=20, the *SS* Activation function boundaries and features scaling is -20 to 20.

Algorithm 1 represents the classical three functions of the *NN* learning process; feed forward, back-propagation, and updating weights.

Algorithm 1: *PNN* learning flow

```
Data:
    $D \in \{\langle x_1,\pi_1\rangle, \{\langle x_2,\pi_2\rangle, ...., \langle x_n,\pi_n\rangle\}$    (4)
Result:
    $\mathscr{L} = \{\lambda_{y_1}, ...., \lambda_{y_n}\}$    (5)
1 Randomly initialize weights $\omega_{i,j} \in \{-0.5, 0.5\}$
2 repeat
3     forall $\langle x_i,\pi_i\rangle \in D$ do
4         $a_i|_{l-1} = \sum_{i=1}^{m} \varphi_{(a_i \cdot \omega_i)|_n}$ // Feedforward
5         Backpropagation()
6         $\omega_{i new} = \omega_{i old} - \eta \cdot \delta_i$ //Update Weights
7 until $\tau_{Avg}$ = 1 or #iterations $\geq 10^6$;
```

Algorithm 2: *PNN* backpropagation

```
8 for l in L-1 to 1 do
9     if l=L-1 then
10        for each i in layer l do
11            $\rho = -6 \cdot (2yt_i - y_i)/m(m^2-1)$ //Spearman error
12    else
13        for each i in layer l do
14            $Err_i = \omega_i \cdot \delta_i$
15    for each i in layer l do
16        $\Delta \cdot \delta_i = Err \cdot \varphi\prime$
```

C. Ranking Loss Function

Two main error functions have been used for label ranking; Kendall τ [?] and Spearman [?]. However, Kendall error function lack continuity and differentiability. Therefore, *PNN* spearman correlation coefficient to measure the ranking between output labels. *Spearman* error derivative is used as a gradient ascent process only for backpropagation and correlation is used as a ranking evaluation function for convergence stopping criteria. τ_{Avg} is the average τ per each label divided by number of instances m, as shown in line 8 of Algorithm 1. *PNN* type B uses Spearman correlation coefficient function and it's derivatives to calculate the gradient ascent and error stopping criteria. *PNN* by *PSS* or *SS* functions takes less calculation time to reach ranking accuracy 100%, As shown in fig **??** in result section where type A has low *RMS* comparing to *Spearman*.

Spearman error function which is represents by the Eq.**??**

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(y_i - yt_i)^2}{n(n^2-1)} \tag{6}$$

where y_i, yt_i, i and m represent rank output value, expected rank value, label index and number of instances, respectively.

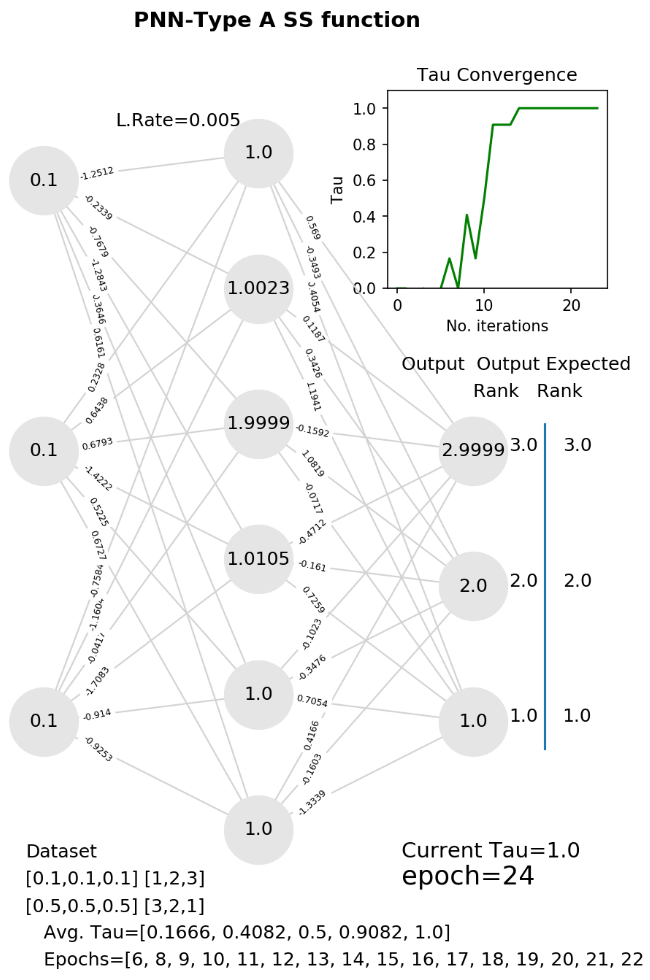
D. PNN Structure

1) Type A: has traditional neural network architecture. It is fully connected single-hidden layer *NN* and multiple-valued neuron has single activation function. The input layer represents the number of features per data instance. The hidden neurons are equal to or greater than the number of output neurons, $H_n \geq \mathscr{L}$, in order to reach error convergence after a finite number of iterations. The output layer represents the labels indexes as neurons, where the labels are displayed in fixed order as shown in Fig. **??**

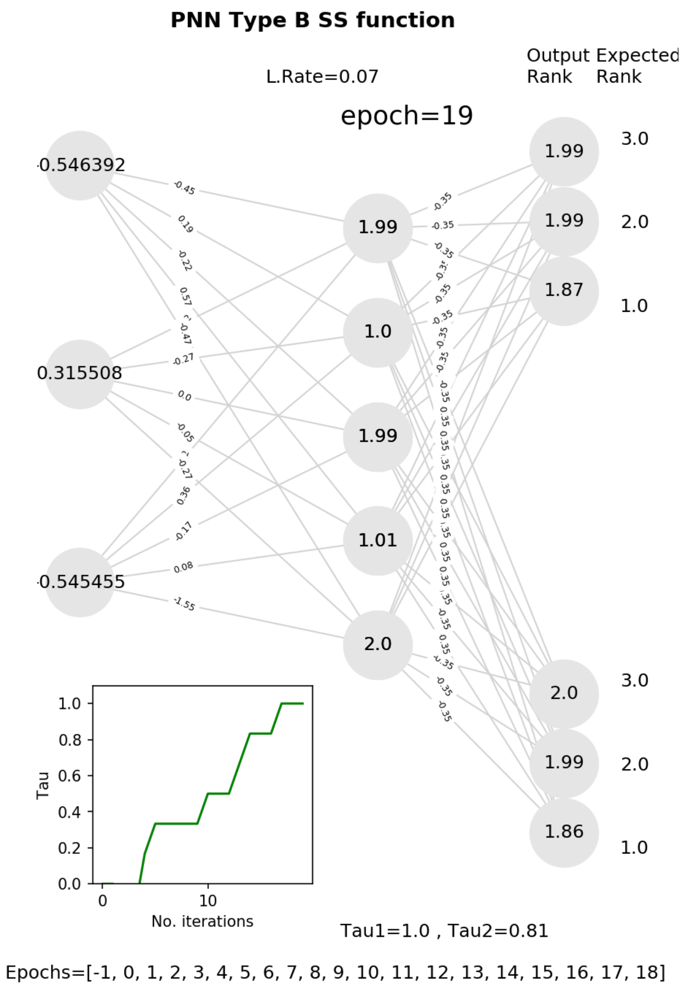
2) Type B: Scaling type A is increasing the number of hidden neurons only because increasing hidden layer does not increase ranking because it doesn't increase arbitrarily complex decision regions and degrees of freedom to solve rank more complex problem. this limitation due to *SS* activation function which doesn't have variation in output values. Therefore, instead of increasing hidden layer , We propose duplication of output layers. this duplication equal to duplication of network from middle to output layer. All these duplicated networks share the weights between input and middle layer. This increasing in number of weights between middle and duplicated output layers increase the number of connection in the system thus increase degree of freedom. This increasing number of weights using the same number of hidden neurons. requires to duplicate activation functions. therefore hidden neuron serve more than one network.

Type B has different architecture to speed up the learning. We introduce hidden neuron multiple activation functions. The duplication of activation functions equal to the duplication of multi label group. each activation function linked to one clone of output labels. as shown in fig **??** (b). Weights are duplicated between middle layer and output layer. This type of enforcement learning reduces the number of iterations and speed up the convergence rate as shown in Fig **??** (b).

3) PNN One middle layer: Both types have one middle layer only. Ranking performance doesn't enhance by increasing number of hidden layers as shown in Fig. **??** This due to the shape of SS and PSS functions. these activation functions don't have smooth slope of variation of values as Relu and Sigmoid to give variance of values. In order to scale the architecture and enhance the performance of the middle layer , We introduce multiple activation functions to middle neurons.


PNN-Type A SS function
L.Rate=0.005
Tau Convergence
Tau
No. iterations
Output Output Expected Rank Rank
Dataset
[0.1,0.1,0.1] [1,2,3]
[0.5,0.5,0.5] [3,2,1]
Avg. Tau=[0.1666, 0.4082, 0.5, 0.9082, 1.0]
Epochs=[6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
Current Tau=1.0
epoch=24

(a)


PNN Type B SS function
L.Rate=0.07
epoch=19
Output Expected Rank Rank
Tau
No. iterations
Tau1=1.0 , Tau2=0.81
Epochs=[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

(b)

Fig. 4: Sample output images of video file demonstrate the evaluation of *PNN* type A and B using *SS* function after 20 iterations in (a) and (b) respectively. [**?**]

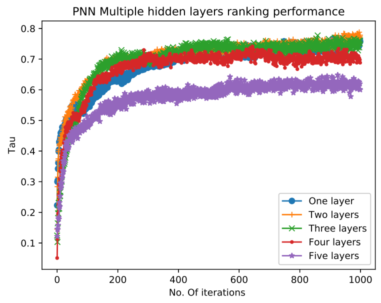


Fig. 5: *PNN* Type A SS ranking performance is decreasing by increasing the number of hidden layers

4) Multiple-Activation functions: In Type B each hidden neuron contain more than one activation function. the activation function has the same type *PSS* or *SS* and has equal number of labels n. middle neuron could have double, triple, or quadratic functions. each function is connected to different output layer which represent a clone of the multi-labels. as shown in the architecture in Fig. **??**.

Fig **??** represents type A and B for a small dataset by reaching ranking correlation = 1 after a small finite number of iterations. as shown in Fig **??** the output labels represents the ranking values. The differential *SS* function helps to reach the convergence after few number of iterations due to Staircase shape which achieve the stability in learning.

In order to map sub-grouping to the label ranking, *PNN* assigns the same ranking values to more than one label. Thus, *NN* structure solves the problem of multiple subgroup ranking where number of ranking values is less the number of output labels, i.e., $\pi = \lambda_d \succ \lambda_b \succ (\lambda_c, \lambda_a)$, where λ_d=1, λ_b=2 and $(\lambda_c \simeq \lambda_a)$ = 3 as shown in Fig. **??**. *PNN* calculates preference values of the output labels based on the bounded *SS* activation function. Each neuron uses activation function in feedforward propagation to calculate the preference number from 1 to n, where n is the number of label classes. During backpropagation, both *SS* and error function are differentiated as shown in Eqs. (10)-(13) and (14)-(15), respectively. The process of feedforward and backpropagation are iterated until the average of Kendall τ coefficient of all data is equal to one (τ_{Avg}=1) or the number of iterations reaches (10^6) as shown in Algorithm 1.

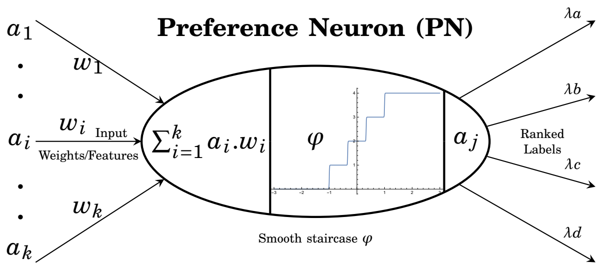


Fig. 6: Preference neuron single function where $\varphi_{n=4}$

where τ, Sgn,y_i, yt_i, i, j and m are Kendall τ coefficient, sign function, output ranking value, expected ranking value, label indexes and number of instances. Incomparable label are excluded from (**??**)

IV. *PNN* BACKPROPAGATION

A. *Backpropagation of Output Layer*

In this step, *SS* activation and the error function are differentiated for every hidden neuron as given in Eq. **??**.

$$\frac{\partial\tau}{\partial\omega_L} = \frac{\partial\tau}{\partial y_L}\cdot\frac{\partial y_L}{\partial y_{aL}}\cdot\frac{\partial y_{aL}}{\partial\omega_L} \tag{7}$$

where y_{aL} is the neuron output before the activation function for middle layer L. For type B, We have multiple output layers and multiple Activation functions as in Eq. **??**

$$\frac{\partial\tau}{\partial\omega_{Lm}} = \frac{\partial\tau}{\partial y_{Lm}}\cdot\frac{\partial y_{Lm}}{\partial y_{aL}}\cdot\frac{\partial y_{aL}}{\partial\omega_{Lm}} \tag{8}$$

Where m is the number of activation functions.

The differentiation of the four labels ranking activation function is represented in Eqs. (**??**)-(**??**).

$$\frac{\partial y_L}{\partial y_{aL}} = -1/2\cdot\sum_{i=0}^{3}(1-tanh(-k\cdot y_{aL}\cdot(k-(2\cdot k\cdot i/n-1))^2) \tag{9}$$

$$\frac{\partial\tau}{\partial\omega_L} = \frac{\partial\tau}{\partial y_L}\cdot\frac{\partial y_{aL}}{\partial\omega_L}\cdot -1/2\cdot\sum_{i=0}^{3}(1-tanh(-k\cdot y_{aL}\cdot(k-(2\cdot k\cdot i/n-1))^2) \tag{10}$$

$$\frac{\partial\tau}{\partial\omega_L} = \frac{\partial\tau}{\partial y_L}\cdot H_L\cdot -1/2\cdot\sum_{i=0}^{3}(1-tanh(-k\cdot y_{aL}\cdot(k-(2\cdot k\cdot i/n-1))^2) \tag{11}$$

The differentiation of Spearman correlations objective function for output layer is given in Eq.**??**

$$\frac{\partial\rho}{\partial y_L} = \frac{12(y_L - y_L y_{tL})}{n(n^2-1)} \tag{12}$$

$$\omega_{L_{new}} = \omega_{L_{old}} - \eta\cdot\frac{\partial\rho}{\partial w_{L_{old}}} \tag{13}$$

where η, ω_L and a are the learning rate, weight of H_1 and output node, respectively as shown in Fig. 4.

$$\omega_{L_{new}} = \omega_{L_{old}} - \eta\cdot\frac{\partial\rho}{\partial y_L}\cdot\frac{\partial y_1}{\partial y_{aL}}\cdot H_L \tag{14}$$

B. *Backpropagation of Middle Layer*

This section shows the calculation of the weights of input neurons of hidden layer using Eqs.(**??**)-(**??**).

$$\frac{\partial\rho_{total}}{\partial w_{L-1}} = \frac{\partial\tau}{\partial H_{L-1}}\cdot\frac{\partial H_{L-1}}{\partial H_{aL-1}}\cdot\frac{\partial H_{aL-1}}{\partial w_{L-1}} \tag{15}$$

where H_a is the hidden neuron output before the activation function.

$$\frac{\partial\rho_{total}}{\partial w_{L-1}} = \frac{\partial\tau_L}{\partial H_L}\cdot\frac{\partial H_{L-1}}{\partial H_{a\cdot L-1}}\cdot\frac{\partial H_{a\cdot L-1}}{\partial\omega_{L-1}} \tag{16}$$

$$\frac{\partial\rho_L}{\partial H_L} = \frac{\partial\rho_L}{\partial y_{aL}}\cdot\frac{\partial y_{aL}}{\partial H_L} \tag{17}$$

$$\omega_{L-1new} = \omega_{L-1old} - \eta.\frac{\partial\rho_{total}}{\partial\omega_{L-1}} \tag{18}$$

$$\omega_{L-1new} = \omega_{L-1old} - \eta.\frac{\partial\rho_L}{\partial H_L} \cdot -1/2\cdot\sum_{i=0}^{3}(1-tanh(-k\cdot y_{aL}\cdot(k-(2\cdot k\cdot i/n-1))^2) \tag{19}$$

V. PROOF OF CONVERGENCE

Ranking convergence is visualized by plotting the value for each output neurons before applying activation function on x-axis and the output after activation function as shown in Fig. 5 - 7 where the point colors represents the actual ranking and y-axis represents the output ranking. By increasing the number of iterations, The ranking accuracy is increased. The τ rate changes according to the shape activation function with the same hyper parameters. For example, *SS* has τ=0.8 in 80 epochs. However, *PSS* takes 30 epochs as shown in in Fig **??**. This τ variation is due to the *PSS* curve smoothness between steps and staircase location on both sides of y-axis. The comparison between SS and PSS by different edge smoothness is shown in Fig. **??**

The output of preference neuron a is obtained from the activation function φ given in Eq. **??**

$$a_j = \varphi(\omega^T\cdot a_i) \tag{20}$$

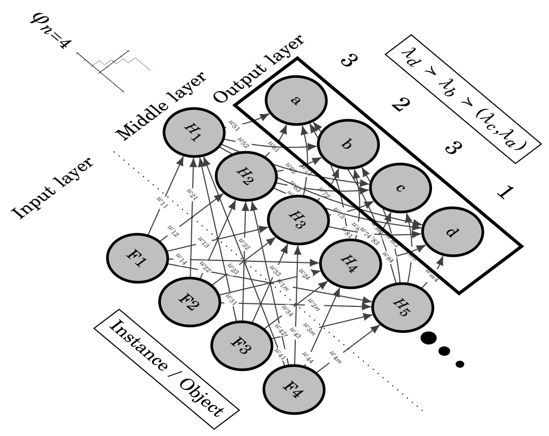
where a_i and a_j are neuron input and output, respectively. The neural output behavior is shown in Eq. **??**

$$\{a_1,t_1\},\{a_2,t_2\},...,\{a_j,t_q\} \tag{21}$$

where each target output t_q is the preference value equal to 1, 2, 3,.., n.

The total inputs to the preference neuron is calculated as the following after neglecting the bias.

$$a_j = \omega^T\cdot a_i = \omega^T\cdot z \tag{22}$$


(a)

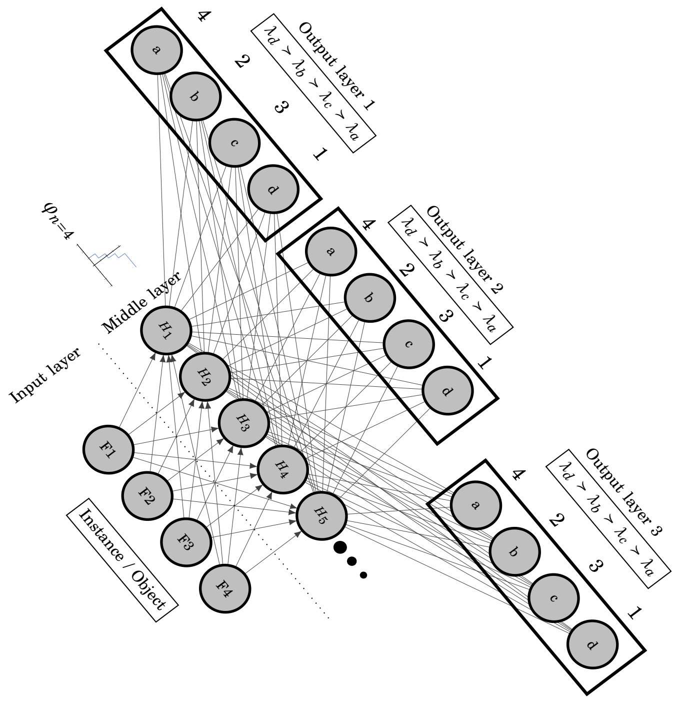

(b)

Fig. 7: *PNN* type A and B architecture in (a) and (b) respectively where $\varphi_{n=4}$, f_{in}=4 and λ_{out}=4.

TABLE I: PNN types.

Type	A	B
Arch.	Full. Connected, FF-NN	Partial Connected, FF-NN
Neuron	Multi. Value, Single function	
Output Layer	Single Layer	Multi. Layer
Activation Fun.	PSS, SS	
Gradient	Ascent Correlation.	
Learning Err. Fun.	Spearman	
Stopping criteria	Spearman	

The weighted vector is given by Eq. (**??**).

$$\omega_{new} = \omega_{old} + E_{err} \cdot z \tag{23}$$

where E_{err} is the ranking error value from 0 to n.

After k iterations for which the weight changes, the learning process is shown in Eq. (**??**).

$$\omega(k) = \omega(k-1) + z'(k-1) \tag{24}$$

The solution weighted vector ω_s ranks all the input Q correctly. $z'(k-1)$ is the appropriate member of the set as shown in Eq. (41)

$$z_1, z_2, z_3, ..., z_Q. \tag{25}$$

To get preference value for 1, 2 and 3, then t_q=1, 2 and 3 as given in Eqs. (**??**)-(**??**).

$$\omega_s^T z_1 > \delta > 0 \tag{26}$$

$$\omega_s^T z_2 > \delta > 1 \tag{27}$$

$$\omega_s^T z_3 > \delta > 2 \tag{28}$$

The objective of the proof of convergence is to find the upper and lower boundaries on the length of the weighted vector. After k iterations, it can be represented as Eq. (**??**)

$$\omega(k) = z'(0) + z'(1) + \cdots + z'(k-1) \tag{29}$$

By taking the inner product of the solution weighted vector ω_s with the weight vector of k iteration, we can obtain Eq. (**??**)

$$\omega_s^T \cdot x(k) = \omega_s^T \cdot z'(0) + \omega_s^T \cdot z'(1) + \cdots + \omega_s^T \cdot z'(k-1) \tag{30}$$

Eqs. (**??**)-(**??**) and (**??**) are substituted as in Eq.**??**

$$\omega_s^T . z'(i) > \delta \tag{31}$$

Therefore,

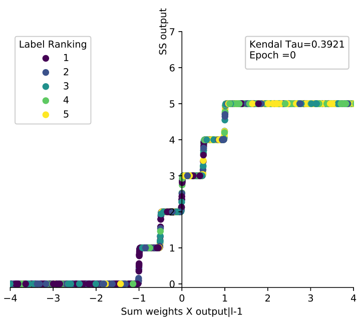
$$\omega_s^T \cdot \omega(k) > k\delta \tag{32}$$

From the Cauchy-Schwarz inequality [**?**]

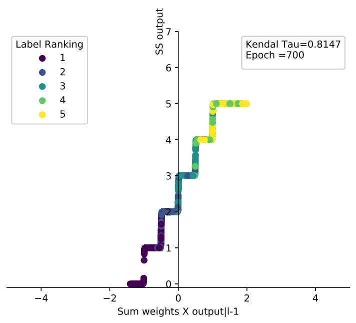
$$(\omega_s^T . \omega(k))^2 \leq \parallel \omega_s \parallel^2 \parallel \omega(k) \parallel^2 \tag{33}$$

where

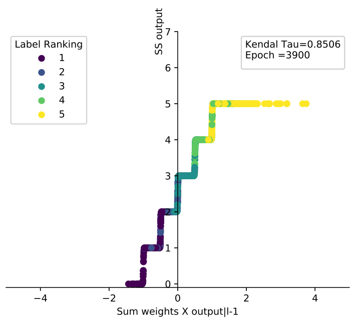
$$\parallel \omega \parallel^2 = \omega^T \omega \tag{34}$$



(a)



(b)



(c)

Fig. 8: Visualizing *PNN* type A convergence of 5 labels using *SS* activation function on stock data set in (a), (b) and (c) after 0, 700 and 3900 epochs respectively.

From Eq. (49) we can put the lower bound on the squared length at iteration k :

$$\parallel \omega(k) \parallel^2 \geq \frac{(\omega_s^T \omega(k))^2}{\parallel \omega_s \parallel^2} > \frac{(k\delta)^2}{\parallel \omega_s \parallel^2} \quad (35)$$

To find an upper bound for the length of weight vector, the change in the length at iteration k is given in Eq. (53)

$$\parallel \omega(k) \parallel^2 = \omega^T(k).\omega(k) \quad (36)$$

$$\parallel \omega(k) \parallel^2 = [\omega(k-1) + z'(k-1)]^T[\omega(k-1) + z'(k-1)] \quad (37)$$

$$\parallel \omega(k) \parallel^2 = \omega^T(k-1)\omega(k-1) + 2\omega^T(k-1)z'(k-1) + z'^T(k-1)z'(k-1) \quad (38)$$

Eq. (52) can be simplified as

$$\parallel \omega(k) \parallel^2 \leq \parallel \omega(k-1) \parallel^2 + \parallel z'(k-1) \parallel^2 \quad (39)$$

Eq. (53) can be repeated for $\parallel \omega(k-1) \parallel^2$, $\parallel \omega(k-2) \parallel^2$, to obtain

$$\parallel \omega(k) \parallel^2 \leq \parallel z'(0) \parallel^2 + \parallel z'(1) \parallel^2 + ... + \parallel z'(k-1) \parallel^2 \quad (40)$$

If $\Pi = max\{\parallel z'(i) \parallel\}$, this upper bound can be simplified to Eq. (58).

$$\parallel \omega(k) \parallel^2 \leq K\Pi \quad (41)$$

The weights only change to a finite number of times because k has upper bound. Therefore, the *NN* learning converges after a finite number of iterations.

VI. Experimental Results

A. Activation Functions Evaluation

PNN is tested on iris data set using four activation functions. *SS*, *PSS*,*Relu*, *Segmoid* and *Tanh*. The experimented *NN* has 1 hidden layer and 50 neurons in the hidden layer and learning rate of 0.05. Fig. **??** shows the convergence after 500 iterations using four activation functions (*SS*, *PSS*, *Sigmoid*, *Relu* and *Tanh*) respectively. We noticed that SS has a stable rate of ranking convergence comparing to *Sigmoid*, *Tanh* and *Relu*. This stability due to the stairstep width that help each point to reach the correct ranking in less number of epochs.

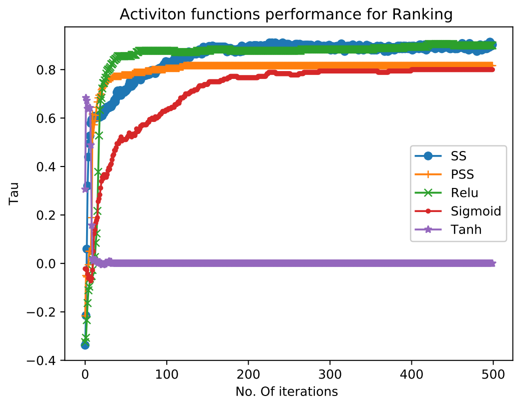


Fig. 9: *PNN* Type A Ranking conversion using the *SS*, *PSS*, *Sigmoid*, *Relu* and *Tanh* activation functions using the iris data set and 500 iterations and learning rate of 0.05 using one hidden layer and 50 hidden neurons.

B. PNN types Evaluation

PNN Type A, B-2, B-3, B-4 are tested on iris Data set using hyper parameters(h.n.=50, L.r=0.07, epochs=100). as shown in Fig. **??** It can be noticed that Type B-4 reach high variance ranking accuracy training model comparing to type A.

C. Objective functions Evaluation

It can be noticed that Type A Spearman reach high variance ranking accuracy training model in 100 iterations comparing type A. However, Type A RMS has lower *RMS* than Spearman as shown in Fig. **??**

D. Data sets

PNN is experimented using three different types of benchmark data sets to evaluate the multi-label ranking performance. The first type of data set focuses on exceptions preference mining [**?**], 'algae' data set is one the first type that highlights indifference preferences problem,

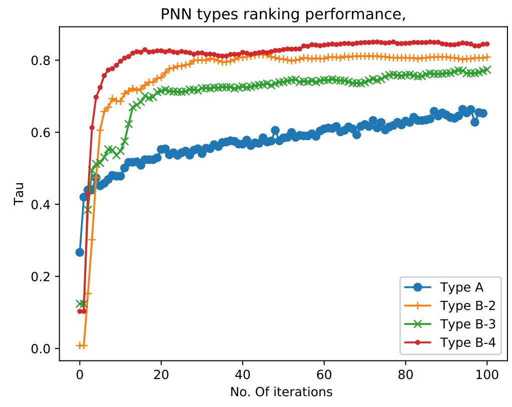


Fig. 10: *PNN* Type A, B-2, B-3, B-4 performance evaluation on iris dataset

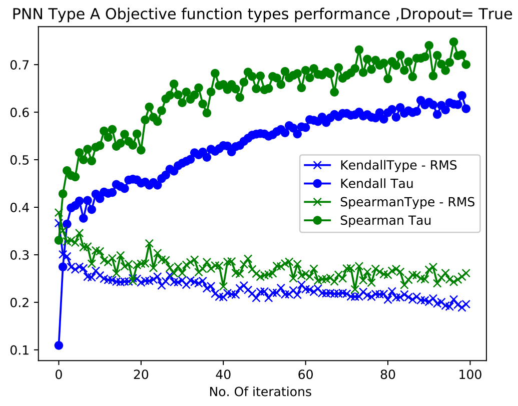


Fig. 11: Objective functions RMS and Spearman in*PNN* type A ain iris dataset

where labels have repeated preference value [?]. German elections 2005, 2009 and modified sushi are considered new and restricted preference data sets. The second type is real-world data related to biological science [?]. The third type of data set is semi-synthetic (*SS*) taken from the *KEBI* Data Repository at the Philipps University of Marburg [?]. All data sets do not have ranking ground truth and all labels have a continuous permutation space of relations between labels. Table I summarizes the main characteristics of the data sets.

E. Results

PNN is evaluated by restricted and non-restricted label ranking data sets. The results are derived using spearman ρ and converted to Kendall τ coefficient for comparison with other approaches. For data validation we use with ten-fold cross validation. to avoid over-fitting problem, Hyper-parameters i.e.(learning rate from 0.001 to 0.5, hidden neuron from 25 to 200 neurons and scaling boundaries from 1 to 50) are chosen within each cross-validation fold by using the best learning rate on each fold and calculating

TABLE II: Three benchmark data sets for label ranking; preference mining [?], semi-synthetic (*SS*) [?] and real-world data sets

type	data set	category	#inst.	#attr.	#lbl.
Mining data	algae	chemical stat.	317	11	4
	german.2005	user pref.	413	29	5
	german.2009	user pref.	413	32	5
	sushi	user pref.	5000	10	10
	top7movies	user pref.	602	7	7
Real data	cold	biology	2,465	23	4
	diau	biology	2,465	24	6
	dtt	biology	2,465	24	4
	heat	biology	2,465	24	6
	spo	biology	2,465	24	11
Semi-Synthesized data	authorship	A	841	70	4
	bodyfat	B	252	7	7
	calhousing	B	20,640	6	5
	cpu-small	B	8192	3	4
	elevators	B	16,599	9	9
	fried	B	40,769	9	5
	glass	A	214	9	6
	housing	B	506	6	6
	iris	A	150	4	3
	pendigits	A	10,992	16	10
	segment	A	2310	3	4
	stock	B	950	5	5
	vehicle	A	846	18	4
	vowel	A	528	10	11
	wine	A	178	13	3
	wisconsin	B	194	16	16

TABLE III: Preference mining ranking performance in terms of Kendall τ coefficient and learning step and number of hidden neurons.

Preference mining exceptions Data			
data set	**Avg.τ**	**l.step**	**#h.n.**
algae	0.651	0.0001	100
german2005	0.89	0.0001	20
german2009	0.78	0.0001	20
sushi	0.69	0.0003	300
top7 movies	0.602	0.004	20

the average τ of ten folds. grid searching is using to get the best hyper parameter.

1) Preference Mining results: The ranking performance of the new preference mining data set is represented in table II. To enhance the ranking performance of the repeated label values of algae data set, a total 250 hidden neurons are used. However, restricted labels ranking data sets of the same type, i.e, (german elections and sushi) did not require high number of hidden neurons and took less computation cost.

2) Restricted preferences results: Table III summarizes *PNN* ranking performance of strict label ranking data sets by learning rate and the total number of hidden neurons. The results are compared with the four methods for label ranking; supervised clustering [?], supervised decision tree [?], multi-layer perceptron label ranking [?] and label ranking tree forest (*LRT*) [?]. Each method's results generated by 10 fold cross validation.The comparison selects only the best approach for each method.

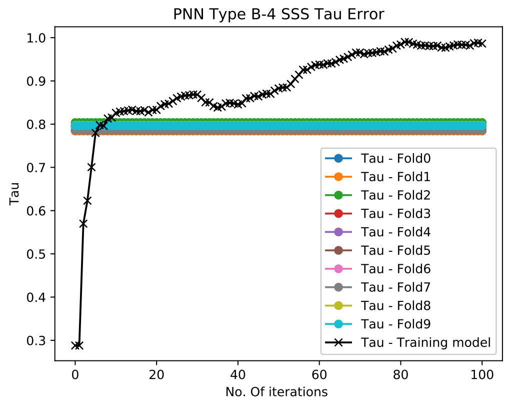


Fig. 12: *PNN* type B results iris evaluation and validation and using Dropout regulation and non-dropout in (a) and (b) respectively.

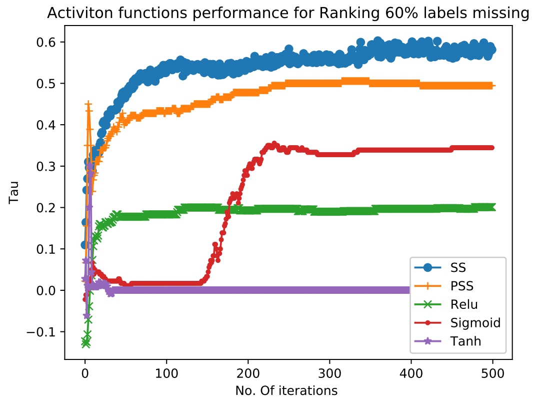


Fig. 13: *PNN* type A Ranking conversion using the *Sigmoid*, *Tanh*,*Relu* and *SS* activation functions using 60% missing labels iris data set and 500 iterations and learning rate of 0.05 using one hidden layer and 50 hidden neurons.

F. PNN Performance

During the experiment, it was found that ranking performance increases with an increase in the number of hidden neurons in hidden layers. All the results are held using single hidden layer with a various number of hidden neurons from (50 to 300) and *SS* activation function. Kendall τ error converges and reaches close to 1 after 2000 iterations as shown in fig 6.

Table IV compares *PNN* with the similar approaches used for multi-label ranking. These approaches are; Decision trees [?], *MLP-LR* [?] and label ranking trees forest *LRT* [?]. In this comparison, we choose the method that have promising results for each approach.

G. Missing Labels Evaluation

PNN is evaluated by removing a random number of labels per each instance. *PNN* marked the missing label as -1; *PNN* neglects error calculation during backpropagation, $\delta = 0$. Thus, the missing label weights remain constants per each learning iteration. Missing label approach is applied to the data set by 20% and 60% of the training data. It is noticed that ranking performance decreases when the number of the missing labels increases.However, *SS* and PSS are more stable convergence than other functions. This evaluation is performed on iris data set as shown in Fig. **??**.

H. PNN Dropout Regularization

We apply dropout as a regularization approach to enhance the neural network performance by reducing overfitting. We dropout weights have probability with less than 0.5. The comparison between dropout and non-dropout of type A and B are shown in Fig. **??** , **??**. the gap between training model and 10 fold cross validations curves has been reduced by using dropout of iris dataset.

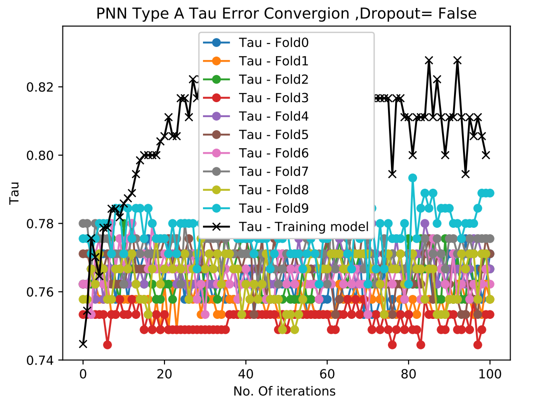


Fig. 14: *PNN* type A results Iris evaluation and validation and using Dropout regulation and non-dropout in (a) and (b) respectively.

I. PSS and SS Evaluation

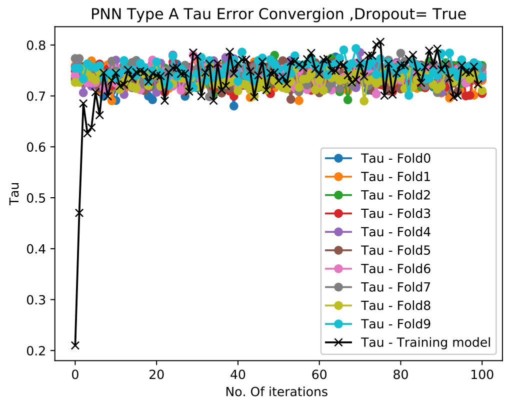
As shown in Fig **??**, *PSS* for A and B reaches the convergence in less number of iterations comparing to *SS*. However, *SS* has lower *RMS* than *PSS* This is due to two reasons: 1- *SS* smoothness between staircase steps is almost 0, This hard smoothness make the results jump to integer ranking values, 2- The symmetry of SS function on x Axis. The *SS* shape handle both positive and negative normalized data. It reduces the number of iterations to reach the correct ranking values.

The biological real world data set was experimented using supervised clustering (*SC*) [?], Table V represents the comparison between *PNN* Type A and supervised clustering on biological real world data in terms of $Loss_{LR}$ as given in Eq. **??**.

$$\tau = 1 - 2 \cdot Loss_{LR} \qquad (42)$$

TABLE IV: *PNN* type A performance comparison with approaches: supervised clustering [?], supervised decision tree [?], multi-layer perceptron label ranking [?] and label ranking tree forest (*LRT*) [?]

Multi Label Ranking Methods						
DS	**S.Clust.**	***DT***	***MLP-LR***	***LRT***	***PNN-A***	***PNN-B***
authorship	0.854	0.936(IBLR)	0.889(LA)	0.882	0.75	0.76
bodyfat	0.09	0.281(CC)	0.075(CA)	0.117	0.5591	0.51
calhousing	0.28	0.351(IBLR)	0.130(SSGA)	0.324	0.271	0.216
cpu-small	0.274	0.50(IBLR)	0.357(CA)	0.447	0.36	0.337
elevators	0.332	0.768(CC)	0.687(LA)	0.760	0.72	0.66
fried	0.176	0.99(CC)	0.660(CA)	0.890	0.81	0.774
glass	0.766	0.883(LRT)	0.818(LA)	0.883	0.8175	0.751
housing	0.246	0.797(LRT)	0.574(CA)	0.797	0.52	0.53
iris	0.814	0.966(IBLR)	0.911(LA)	0.947	0.817	0.867
pendigits	0.422	0.944(IBLR)	0.752(CA)	0.935	0.81	0.87
segment	0.572	0.959(IBLR)	0.842(CA)	0.949	0.916	0.923
stock	0.566	0.927(IBLR)	0.745(CA)	0.895	0.714	0.7374
vehicle	0.738	0.862(IBLR)	0.801(LA)	0.827	0.654	0.7204
vowel	0.49	0.90(IBLR)	0.545(CA)	0.794	0.65	0.631
wine	0.898	0.949(IBLR)	0.931(LA)	0.882	0.90	0.918
wisconsin	0.09	0.629(CC)	0.235(CA)	0.343	0.612	0.671
Average	0.475	0.79	0.621	0.730	0.6735	0.692



where τ is Kendall τ ranking error and $Loss_{LR}$ is the ranking loss function.

We use the SSS function with 16 steps to rank Wisconsin data set that has 16 labels. By increasing the number of steps in the interval and scaling up the features between -100 and 100, The step width is small. In order to enhance the ranking performance data set has many multi-labels, The number of hidden neurons is increased in order to exceed τ ranking 0.5 as shown in Fig.7.

Fig.9 compares the *Tau* convergence rate between *SS* and *PSS* activation functions using iris data set. The learning step is 0.007 and number of iterations is 5000. We noticed the *SS* performs better in fewer iterations than *PSS* using the same hyper parameters. This is due to the pyramid shape where ranking values can be reached on both sides of y-axis. Thus, it minimize the iterations of back-propagation and weight updating to reach the correct ranking value.

J. Discussion and Future Work

It can be noticed from table III that *PNN* outperforms on *SS* data sets with τ_{Avg}=0.825, whereas other methods such as, supervised clustering, decision tree, *MLP-ranker* and *LRT*, have results τ_{Avg}= 0.79, 0.73, 0.62, 0.475, respectively. Also, the performance of *PNN* is almost 50% better than supervised clustering in terms of ranking loss function $Loss_{LR}$ on biological real world data set as shown in table V. *PNN* increases the number of iterations to enhance the ranking performance of missing labels of iris data set as shown in Fig. 7.

TABLE V: Comparison between *PNN* type A and supervised clustered on biological real world data in terms of $Loss_{LR}$

Biological real world data		
DS	**S.Clustering**	***PNN***
cold	0.198	0.11
diau	0.304	0.255
dtt	0.124	0.01
heat	0.072	0.013
spo	0.118	0.014
Average	0.1632	0.0804

The superiority of *PNN* for ranking is encoding the multi-label preference relation to numeric values and rank the output labels simultaneously. *PNN* could be used to solve new preference mining problems. One of these problems is incomparability between labels, where Label ranking has incomparable relation $\perp$, i.e., ranking space $(\lambda_a \succ \lambda_b \perp \lambda_c)$ is encoded to (1, 2, -1) and $(\lambda_a \succ \lambda_b) \perp (\lambda_c \succ \lambda_d)$ is encoded to (1, 2, -1, -2). *PNN* could be used to solve new problem of non-strict partial orders ranking, i.e., ranking space $(\lambda_a \succ \lambda_b \succeq \lambda_c)$ is encoded to (1, 2, 3) or (1, 2, 2). Future research may focus on modifying *PNN* architecture by adding bias and solving problems of extreme multi-label ranking.

VII. Conclusion

This paper proposed a novel method to rank a complete multi-label space. The preference neural network is native

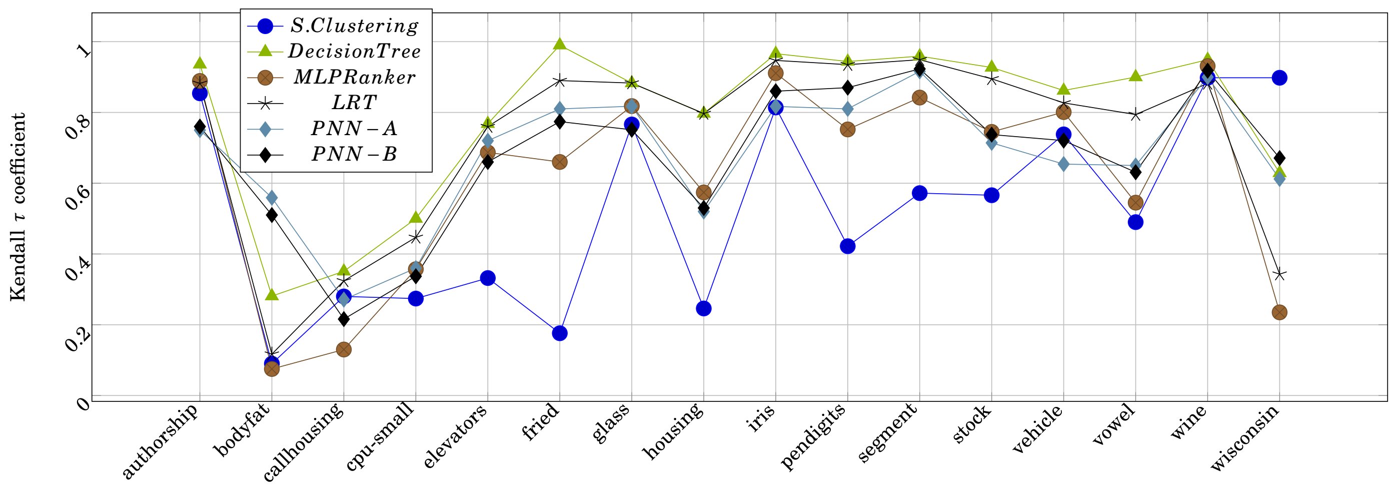


Fig. 15: Ranking performance comparison of *PNN* with other approaches.

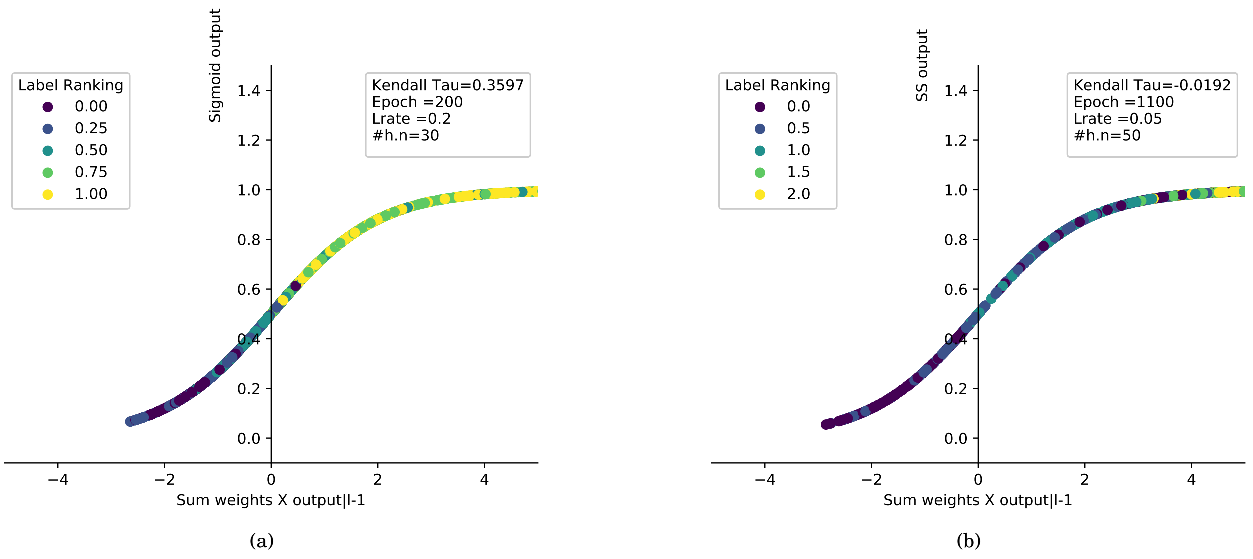


Fig. 16: Ranking convergence of 5 labels using *Sigmoid* activation function on iris data set in (a), (b) after 200, 1100 epochs respectively.

ranker *NN* uses *SS* to rank the multi-label per instance. The novelty of this neural network is overall error is measured by ranking the preference value not by classification error for each neuron. Thus, it takes less computational time with single hidden layer. It indexing multi-labels as output neurons with preference values and proposing new activation function for ranking. The neuron output structure can be mapped to integer ranking value; thus, *PNN* solves sub grouping ranking problems by assigning the rank value to more than one output index. *PNN* is implemented using python programming language and activation functions are modeled using wolframe software [**?**]. the videos of *PNN* types on small dataset is located [**?**]

Acknowledgement

This work was supported in part by the Australian Research Council (ARC) under discovery grant DP180100670 and DP180100656.

References

[1] J. Fürnkranz and E. Hüllermeier, “Preference Learning: An Introduction in Preference Learning”, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1-17.
[2] R. Brafman and C. Domshlak. “Preference handling - an introductory tutorial”. AI Magazine, 30(1): 58-86, 2009.
[3] G. Adomavicius and A. Tuzhilin. “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions.” IEEE Transactions on Knowledge and Data Engineering, 17(6):734-749, 2005.
[4] M. Montaner, B. Lopez and J.L. De La Rosa, “A Taxonomy of Recommender Agents on the Internet.” Artif. Intell. Rev. 19(4), 285-330 (2003)

TABLE VI: *PNN* ranking performance in terms of τ coefficient, learning step and number of hidden neurons.

type	data set	*PNN*	Avg. τ	l.step	#h.n.	#Iterations.	Scaling.
Real data	cold	A	0.4	0.004	10	2000	-1:1
		B	0.42	0.004	10	2000	-1:1
	diau	A	0.466	0.0001	20	2000	-1:1
		B	0.41	0.0001	20	2000	-1:1
	dtt	A	0.60	0.0005	10	2000	-1:1
		B	0.58	0.0005	10	2000	-1:1
	heat	A	0.876	0.01	10	2000	-1:1
		B	0.806	0.004	10	2000	-1:1
	spo	A	0.8	0.0009	20	2000	-1:1
		B	0.75	0.0007	20	2000	-1:1
Semi-Synthesized data	authorship	A	0.75	0.05	30	2000	-1:1
		B	0.76	0.04	30	2000	-1:1
	bodyfat	A	0.559	0.007	14	2000	-100:100
		B	0.51	0.007	14	2000	-100:100
	calhousing	A	0.271	0.0001	20	2000	-1:1
		B	0.212	0.0001	20	2000	-1:1
	cpu-small	A	0.36	0.0039	50	2000	-1:1
		B	0.337	0.007	50	2000	-1:1
	elevators	A	0.72	0.007	20	2000	-1:1
		B	0.66	0.0001	20	2000	-1:1
	fried	A	0.81	0.01	100	2000	-1:1
		B	0.774	0.0001	100	2000	-1:1
	glass	A	0.81	0.0001	50	2000	-1:1
		B	0.75	0.0001	50	2000	-1:1
	housing	A	0.52	0.0001	25	2000	-20:20
		B	0.53	0.001	25	2000	-20:20
	iris	A	0.81	0.0019	15	2000	-1:1
		B	0.867	0.0019	15	2000	-1:1
	pendigits	A	0.86	0.09	100	50	-10:10
		B	0.817	0.0001	50	50	-1:1
	segment	A	0.914	0.0005	20	2000	-1:1
		B	0.924	0.0005	20	2000	-1:1
	stock	A	0.71	0.0001	50	2000	-1:1
		B	0.73	0.0003	50	2000	-1:1
	vehicle	A	0.65	0.0001	100	2000	-100:100
		B	0.72	0.0001	100	2000	-100:100
	vowel	A	0.65	0.0005	22	2000	-50:50
		B	0.63	0.0001,0.0005	22	2000	-50:50
	wine	A	0.9	0.0009	50	2000	-1:1
		B	0.91	0.0009	50	2000	-1:1
	wisconsin	A	0.61	0.0007	100	2000	-10:10
		B	0.671	0.05	100	2000	-10:10

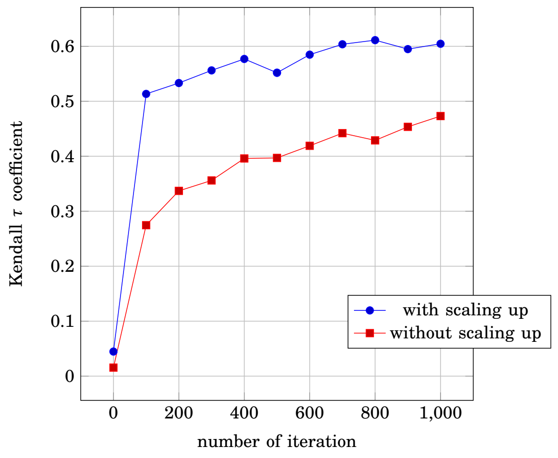


Fig. 17: Performance of ranking wisconsine data set has 16 labels with and without scaling up approach.

[5] F. Aiolli,"A preference model for structured supervised learning tasks", in Proceedings of the IEEE International Conference on Data Mining (ICDM) (2005), pp. 557-560

[6] K. Crammer and Y. Singer, "Pranking with ranking", in Advances in Neural Information Processing Systems (NIPS) (2002), pp. 641-647

[7] C. Burges et al., "Learning to rank using gradient descent," presented at the Proceedings of the 22nd international conference on Machine learning, Bonn, Germany, 2005.

[8] Y. Zhou, Y. Liu, J. Yang, X. He, and L. Liu, "A Taxonomy of Label Ranking Algorithms," JCP, vol. 9, pp. 557-565, 2014.

[9] J. Furnkranz and E. Hüllermeier, "Pairwise Preference Learning and Ranking," in Machine Learning: ECML 2003, Berlin, Heidelberg, 2003, pp. 145-156: Springer Berlin Heidelberg.

[10] J. Fürnkranz and E. Hüllermeier, "Preference Learning," Springer-Verlag, 2010.

[11] S. Har-Peled, D. Roth, and D. Zimak, "Constraint classification for multiclass classification and ranking," presented at the Proceedings of the 15th International Conference on Neural Information Processing Systems, 2002.

[12] P. L. H. Yu, W. M. Wan, and P. H. Lee, "Decision Tree Modeling for Ranking Data," in Preference Learning, J. Furnkranz and E. Hüllermeier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 83-106.

[13] C. R. de Sá, C. Soares, A. M. Jorge, P. J. Azevedo, and J. P. da Costa, "Mining association rules for label ranking" in PAKDD (2), 2011, pp. 432-443.

[14] A. G. e. Ivakhnenko, V. G. v. Lapa, S. United, and S. Joint Publications Research, Cybernetic predicting devices. New York: CCM Information Corp., 1965.

[15] A. G. e. Ivakhnenko, V. G. Lapa, and R. N. McDonough, "Cybernetics and forecasting techniques." New York: American Elsevier, 1967.

[16] A. G. Ivakhnenko, "The Group Method of Data Handling-A Rival of

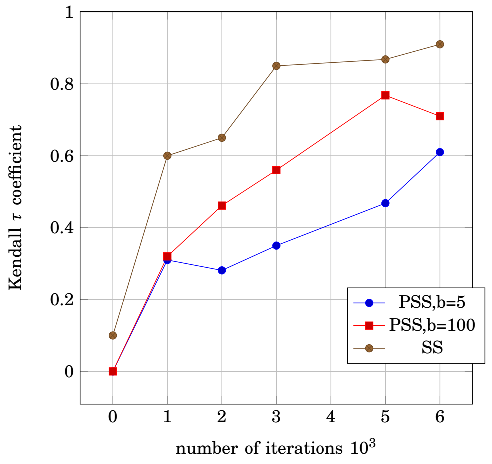


Fig. 18: Ranking comparison of wisconsin data set using SS and *PSS* b = 5, 100 and *SS* respectively. Where b is the step curve value Activation functions where *PSS* with sharp edges gives better performance.

the Method of Stochastic Approximation," Soviet Automatic Control, Vol. 1, No. 3, 1968, pp. 43-55.

[17] R. Isermann, S. Ernst, and O. Nelles,"Identification with Dynamic Neural Networks - Architectures, Comparisons, Applications," IFAC Proceedings Volumes, vol. 30, no. 11, pp. 947-972, 1997/07/01/ 1997.

[18] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85-117, 2015/01/01/ 2015.

[19] X. Wang, T. Huang, and X. Liu, "Handwritten Character Recognition Based on BP Neural Network," in 2009 Third International Conference on Genetic and Evolutionary Computing, 2009, pp. 520-524.

[20] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 2, pp. 513-529, 2012.

[21] E. Hüllermeier, J. Furnkranz, W. Cheng, K. Brinker, "Label ranking by learning pairwise preferences". Artif. Intell. 178, 1897-1916 2008.

[22] O. Dekel, D.Manning, Y. Singer, "Log-linear models for label ranking", in Advances in Neural Information Processing Systems, vol. 16 2003.

[23] Y. H. Jung and A. Tewari, "Online Boosting Algorithms for Multi-label Ranking," ed. Ithaca: Cornell University Library, arXiv.org, 2018.

[24] C. R. de Sá, W. Duivesteijn, P. Azevedo, A. M. Jorge, C. Soares, and A. Knobbe, "Discovering a taste for the unusual: exceptional models for preference mining," Machine Learning, vol. 107, no. 11, pp. 1775-1807, 2018/11/01 2018.

[25] http://dx.doi.org/10.17632/3mv94c8jpc.2

[26] S. Har-Peled, D. Roth, D. Zimak, Constraint classification: A new approach to multiclass classification, in Proceedings of the Thirteenth International Conference on Algorithmic Learning, Theory 2002.

[27] M. Grbovic, N. Djuric, S. Guo, and S. Vucetic, "Supervised clustering of label ranking data using label preference information," Machine Learning, vol. 93, no. 2, pp. 191-225, 2013/11/01 2013.

[28] W. Cheng et al., "Decision tree and instance-based learning for label ranking," presented at the Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Quebec, Canada, 2009.

[29] G. Ribeiro, W. Duivesteijn, C. Soares, and A. Knobbe,"Multilayer perceptron for label ranking," presented at the Proceedings of the 22nd international conference on Artificial Neural Networks and Machine Learning - Volume Part II, Lausanne, Switzerland, 2012.

[30] W. Cheng and E. Hüllermeier, "Instance-based label ranking using the mallows model," in ECCBR Workshops, 2008, pp. 143-157.

[31] M. Grbovic, N. Djuric, and S. Vucetic, "Learning from Pairwise Preference Data using Gaussian Mixture Mosdel," 2014.

[32] M. G. Kendall, "Rank Correlation Methods." London, England: Griffin, 1970.

[33] C. Spearman, "The proof and measurement of association between two things,"The American Journal of Psychology, vol. 15, no. 1, pp. 72-101, 1904.

[34] A. Aiguzhinov, C. Soares, and A. P. Serra, "A Similarity-Based Adaptation of Naive Bayes for Label Ranking: Application to the Metalearning Problem of Algorithm Recommendation,"in Discovery Science, Berlin, Heidelberg, 2010, pp. 16-26: Springer Berlin Heidelberg.

[35] C. R. de Sá, C. Soares, A. Knobbe, and P. Cortez, "Label Ranking Forests," Expert Systems, vol. 34, no. 1, 2017.

[36] Q. Song, H. Jin, X. Huang, and X. Hu, "Multi-Label Adversarial Perturbations," ed. Ithaca: Cornell University Library, arXiv.org, 2019.

[37] Y. Yan, Y. Wang, G. Wen-Chao, Z. Bo-Wen, C. Yang, and Y. Xu-Cheng, "LSTM 2: Multi-Label Ranking for Document Classification," (in English), Neural Processing Letters, vol. 47, no. 1, pp. 117-138, 2018 2018-03-03 2018.

[38] Y. Zhou and G. Qiu, "Random forest for label ranking," (in English), Expert Systems with Applications, vol. 112, p. 99, 2018 Dec 01 2018-10-05 2018.

[39] B. Guo, C. Hou, J. Shan, and D. Yi, "Low Rank Multi-Label Classification with Missing Labels," in 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 417-422.

[40] V. Bergelson, "Part V: Theorems and Problems - 09. Ergodic Theorems." Princeton: Princeton University Press, 2008, pp. 3-691.

[41] Wolfram Research, Inc., Mathematica, Version 12.0, Champaign, IL (2019).

[42] A. Elgharabawy. "PNN Convergence performance." Video File, Feb. 26, 2020 [Video file]. Available: https://drive.google.com/drive/folders/1yxuqYoQ3Kiuch-2sLeVe2ocMj12QVsRM?usp=sharing [Accessed: Jan. 28, 2020].

[43] G. Bologna, "Rule extraction from a multilayer perceptron with staircase activation functions," in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 2000, vol. 3, pp. 419-424 vol.3.

[44] C. Moraga and R. Heider, ""New lamps for old!" (generalized multiple-valued neurons)," in Proceedings 1999 29th IEEE International Symposium on Multiple-Valued Logic (Cat. No.99CB36329), 1999, pp. 36-41.