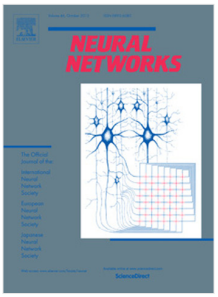
Accepted Manuscript



An improved robust heteroscedastic probabilistic neural network based trust prediction approach for cloud service selection

Nivethitha Somu, Gauthama Raman M.R., Kalpana V., Kannan Kirthivasan, Shankar Sriram V.S.

PII: S0893-6080(18)30225-9
DOI: https://doi.org/10.1016/j.neunet.2018.08.005
Reference: NN 4009

To appear in: *Neural Networks*

Received date : 21 December 2017
Revised date : 12 July 2018
Accepted date : 3 August 2018

Please cite this article as: Somu, N., Gauthama Raman M.R, Gauthama Raman M.R., Kalpana V, Kalpana V., Kirthivasan, K., Shankar Sriram V.S., Shankar Sriram V.S.., An improved robust heteroscedastic probabilistic neural network based trust prediction approach for cloud service selection. *Neural Networks* (2018), https://doi.org/10.1016/j.neunet.2018.08.005

Highlights

- ✓ An ***Improved Robust Heteroscedastic Probabilistic Neural Network (HC-RHRPNN)*** is presented for ***trust prediction in cloud environments***
- ✓ HC-RHRPNN employs ***hypergraph coarsening based dimensionality reduction technique*** for the ***identification of informative samples***
- ✓ HC-RHRPNN was evaluated using ***QWS dataset*** in terms of ***classification accuracy, precision, recall, and F-Score***
- ✓ ***One way ANOVA statistical test*** was performed to prove the dominance of HC-RHRPNN

*Title Page (With all author details listed)

Title Page

Authors:

Nivethitha Somu[1], Gauthama Raman M R[1], Kalpana V[1], Kannan Krithivasan[2], Shankar Sriram V S[1*]

Email id:

nivethitha@sastra.ac.in[1], gauthamaraman_mr@sastra.ac.in[1], kalpana@cse.sastra.edu[1], kkannan@maths.sastra.edu[2], sriram@it.sastra.edu[1]

Manuscript Title:

An Improved Robust Heteroscedastic Probabilistic Neural Network based Trust Prediction Approach for Cloud Service Selection

Affiliation:

[1] *Centre for Information Super Highway (CISH),* School of Computing, SASTRA Deemed University, Thanjavur, Tamil Nadu, India

[2] *Discrete Mathematics Research Laboratory (DMRL),* Department of Mathematics, SASTRA Deemed University, Thanjavur, Tamil Nadu, India

Corresponding Author:

Prof. Shankar Sriram V S

Email id: sriram@it.sastra.edu

Telephone: +91 4362 264101 (Extn: 2323)

Fax: +91 4362 264120

Acknowledgements

This work was supported by The Department of Science and Technology, India; The Council for Scientific and Industrial Research, India; TATA Realty - SASTRA Srinivasa Ramanujan Research Cell , India (Grant No: DST/INSPIRE Fellowship/2013/963, CSIR-SRF Fellowship/143404/2K15/1, MRT/2017/000155, SR/FST/MSI-107/2015 and SR/FST/ETI-349/2013).

An Improved Robust Heteroscedastic Probabilistic Neural Network based Trust Prediction Approach for Cloud Service Selection

Nivethitha Somu[1], Gauthama Raman M R[1], Kalpana V[1], Kannan Kirthivasan[2], Shankar Sriram V S[1*]

[1] Centre for Information Super Highway (CISH), School of Computing,

[2] Discrete Mathematics Research Laboratory (DMRL), Department of Mathematics,

SASTRA Deemed University, Thanjavur, Tamil Nadu, India

Email id: nivethitha@sastra.ac.in[1], gauthamaraman_mr@sastra.ac.in[1], kalpana@cse.sastra.edu[1], kkannan@maths.sastra.edu[2], sriram@it.sastra.edu[1]*

Abstract

Trustworthiness is a comprehensive quality metric which is used to assess the quality of the services in service-oriented environments. However, trust prediction of cloud services based on the multi-faceted Quality of Service (QoS) attributes is a challenging task due to the complicated and non-linear relationships between the QoS values and the corresponding trust result. Recent research works reveal the significance of Artificial Neural Network (ANN) and its variants in providing a reasonable degree of success in trust prediction problems. However, the challenges with respect to weight assignment, training time and kernel functions make ANN and its variants under continuous advancements. Hence, this work presents a novel multi-level Hypergraph Coarsening based Robust Heteroscedastic Probabilistic Neural Network (HC-RHRPNN) to predict trustworthiness of cloud services to build high-quality service applications. HC-RHRPNN employs hypergraph coarsening to identify the informative samples, which were then used to train HRPNN to improve its prediction accuracy and minimize the runtime. The performance of HC-RHRPNN was evaluated using Quality of Web Service (QWS) dataset, a public QoS dataset in terms of classifier accuracy, precision, recall, and F-Score.

Keywords: Cloud service selection; Quality of service; Trust prediction; Hypergraph; Heteroscedastic probabilistic neural network.

1. Introduction

Cloud computing, an efficient, and economic business paradigm has attracted a wide range of organizations as it enables the users to access on-demand resources as a service ('XaaS'-Something as a Service) over the internet in a 'Pay-As-You-Use' fashion (Sosinsky, 2010). The increasing popularity of cloud computing has resulted in the proliferation of many Cloud Service Providers (CSPs) and functionally equivalent cloud services. On the other end, the Cloud Users (CU) lack appropriate information and benchmarks to evaluate these services based on their preferences and CSPs provisions (Ali Sunyaev, 2013). In addition, the trade-off between the functional and non-functional Quality of Service (QoS) requirements hardens the identification of appropriate and trustworthy CSPs who can satisfy the users' unique QoS requirements. Thereby, the presence of a wide range of cloud-based entities (service providers, users, applications and unique demands) has

provoked the research communities towards the development of cloud service selection models based on several approaches like multi-criteria decision making (multiple attributes and interrelations among them), optimization, logic, description, and trust (Ma, Zhu, Hu, Li, & Tang, 2017; Qu, 2016; Sun, Dong, Hussain, Hussain, & Chang, 2014).

Recent literature reveals the significance of trust based cloud service selection models for the cloud service selection problem in service-oriented environments (Akshya Kaveri, Gireesha, Somu, Gauthama Raman, & Shankar Sriram, 2017; Somu, M.R., Krithivasan, & V.S., 2018; Tang, Dai, Liu, & Chen, 2017). The trustworthiness or quality of a cloud service is reflected by its non-functional (QoS monitoring) and functional attributes (users' feedbacks). Generally, trust assessments were carried out based on monitored QoS values, due to the practical difficulty in obtaining reliable and complete users' feedbacks (Mao, Lin, Xu, & He, 2017). However, the dynamic nature of the cloud environment (unpredictable nature of QoS) and the emergence of new cloud services based on the unique requirements of the user (complex trust evaluation mechanisms) complicates the cloud service selection problem. Trust prediction, a classification problem is a plausible solution for the above challenges and solve issues related to data sparsity and 'cold start problem' in service selection, composition and recommendation models (Chen, Shen, Li, & You, 2017). Specifically, trust prediction techniques can be applied in a scenario where the trustworthiness of a new entity (service) needs to be evaluated with minimal knowledge on the characteristics of the entity. Realization of the importance of trust-based service evaluation has led to the development of several trust and QoS prediction models in web service selection, cloud service selection, recommender systems, pervasive environments, and social networks (Table 1).

Table 1: Related Works

Author	Technique	Dataset	Metric	Application
(Fu, Hu, & Zhang, 2008)	Bayesian Network	Simulation – Market model for 48 grid service	-	Grid services
(Nguyen, Hien, Weiliang, & Jian, 2010)		Simulation – 3 customers and 5 web services	-	Web services
(Mehdi, Bouguila, & Bentahar, 2013)	• Bayesian Network • Multinomial generalized Dirichlet distribution	Simulation – 4 service, 5 quality metrics, and QWS service classification	-	Web services
(Mohanty, Ravi, & Patra, 2010)	Back Propagation Neural Network (BPNN), Probabilistic Neural Network (PNN), Group Method of Data Handling (GMDH), Classification and	QWS dataset	Classifier accuracy	Web services

	Regression Trees (CART), TreeNet, Support Vector Machine (SVM), and ID3 decision tree (J48)			
(Mashinchi, Li, Orgun, & Wang, 2011)	Fuzzy Linear Regression Analysis (FLRA)	QWS dataset	Classifier accuracy and standard deviation	Web services
(Zolfaghar & Aghaie, 2012)	CRoss-Industry Standard Process (CRISP)	Epinions	Max Profit, overall accuracy, area under the ROC curve, build time, number of fields, prediction accuracy, precision, recall, and F-measure	Social web applications
(Huang, Nie, Huang, Lei, & Ding, 2013)	• Rank-k matrix • Ancillary variables and Augmented Lagrangian Multiplier (ALM)	Epinions, Wikipedia, and Slashdot	Root mean square error and mean absolute error	Social networks
(Raj & Babu, 2017)	Probabilistic reputation feature model	Epinions, Wikipedia, and Slashdot	Overall accuracy, F1 score, and area under the ROC	Social networks
(Zhang, Wu, & Liu, 2016)	Cluster-level trust prediction based on multi-modal social network (CTPMSN).	Epinions, Douban, and Flixster	Average recall and normalized discounted cumulative gain	Social networks
(David Nuñez-Gonzalez, Graña, & Apolloni, 2015)	Trust prediction system based on reputation features	Epinions, and Wikipedia	Average accuracy, recall, and precision	Social networks
(Yu & Huang, 2016)	QoS prediction approach based on time and location-aware collaborative filtering	WSDream QoS dataset	Mean absolute error, and normalized mean absolute error	Web services
(Xu, Yin, Deng, N. Xiong, & Huang, 2016)	An ensemble model for context-aware QoS prediction		Mean absolute error, and root mean squared error	
(Wu, Yue, Li, Zhang, & Hsu, 2017)	QoS prediction using context-sensitive matrix factorization			Cloud services
(Su, Xiao, Liu, Zhang, & Zhang, 2017)	Trust-Aware QoS Prediction approach (TAP)	WSDream QoS dataset	Mean absolute error, and normalized mean absolute error	Web service recommend ation
(Hui Fang, Guo, & Zhang, 2015)	Trust prediction framework – logistic regression	Epinions, FilmTrust, and Flixster	Root mean square error and mean absolute error	Recommen der systems

(Luo, Lv, Li, & Chen, 2015)	• Adaptive dynamic programming • Fuzzy neural networks	WSDream QoS dataset	Mean absolute error and mean absolute percentage error	Cloud services
(Mao et al., 2017)	PSO driven neural networks	QWS dataset	Prediction precision	Cloud services

Further, researchers have extensively studied various statistical and machine learning techniques (Artificial Neural Networks (ANN), Bayesian networks, evidential reasoning, fuzzy, game theory, etc.) for trust prediction (Fu et al., 2008; Mashinchi et al., 2011; Yahyaoui, 2012). Among these, ANN and its variants like Back Propagation Neural Network (BPNN), Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) etc. have proven itself in trust value prediction due to its self-learning ability in modelling complex and arbitrary relationships among the QoS attributes and their trust values in a cloud service selection model (Han & Cho, 2005; Mohanty et al., 2010; M. R. G. Raman, Somu, Kirthivasan, & Sriram, 2017). However, the classical neural network architectures suffer from few intrinsic issues such as (i) identification of optimal connection weights and learning rate, (ii) network stability, (iii) overfitting, and (iv) training time when applied to massive, online, and ill-conditioned dataset (Kuremoto, Kimura, Kobayashi, & Obayashi, 2014). In order to overcome these issues, D.F.Specht proposed Probabilistic Neural Network (PNN), a feedforward neural network derived from a statistical algorithm named Kernel Fischer discriminant analysis and Bayesian network (D. F. Specht, 1990). The independent nature of PNN with respect to the weighting factor and training time makes PNN, an attractive model for various classification and pattern recognition problems in the field of bioinformatics, network security, image processing, etc. In the basic version of PNN, the mixing coefficients and the common variance of the kernel functions were computed from the entire set of training samples (Sivakumar & Kannan, 2009). However, in a real-world scenario, the data (QoS values) obtained from multiple heterogeneous data sources which are massive and ill-conditioned in nature affect the performance of PNN in terms of training time and prediction accuracy. This emphasizes the need to employ a set of kernel functions to process the real-time data (QoS values) obtained from multiple sources without compromising the performance of PNN.

Hence, in this work, we present Hypergraph Coarsening based Robust Heteroscedastic PNN (HC-RHRPNN), an enhanced version of HRPNN for trust prediction in cloud environments. The novelty of HC-RHRPNN lies in two modules, namely (i) ***Pruning:*** Multi-level hypergraph coarsening by maximum edge matching was employed for the identification of informative samples to minimize the training time and enhance the performance of the learning model, and (ii) ***Training:*** The informative samples identified by the pruning module were used for training HRPNN to guarantee high prediction rate. The effectiveness of HC-RHRPNN was evaluated with extensive experiments on the Quality of Web Service (QWS), a public real-time QoS dataset in terms of classifier accuracy, precision, recall and F-Score (Al-Masri & Mahmoud, 2007).

The rest of the paper is organized as follows. Section 2 provides an insight into basic terminologies of trust prediction, PNN, and hypergraph. Section 3 introduces HC-RHRPNN, a multilevel hypergraph coarsening based probabilistic neural network for trust prediction in cloud environments. Section 4 discusses the performance evaluation of HC-RHRPNN over the existing classifiers in terms of various quality metrics. Section 5 concludes the paper.

2. Materials and Methods

2.1 Basic Definitions

Definition 2.1.1 (*QoS Attributes*): Consider $'m'$ number of Cloud Service Providers (CSPs), $CSP_i = (CSP_1, CSP_2, \dots, CSP_m)$ that provides various cloud services to a diverse set of cloud users. The QoS attributes of the CSPs are represented by an n tuple vector (CSMIC-SMI QoS attribute) $Q_{ij} = \{Q_{i1}, Q_{i2}, \dots, Q_{mn}\}, \forall j = \{1,2, \dots, n\}$.

Definition 2.1.2 *(Trust Rate)*: Trust rate, a comprehensive metric to evaluate the trustworthiness of a CSP based on the performance of n QoS attributes. The trust rate of the CSPs is represented by $TV_i = (TV_1, TV_2, \dots, TV_m)$. The complete set of QoS and trust record of each CSP is represented as two tuple vector $QR_i = < (Q_{ij}), (TV_i) >= \{Q_{i1}, Q_{i2}, \dots, Q_{mn}, TV_i\}, \forall j = \{1,2, \dots, n\}$.

Definition 2.1.3 *(Trust Rate Dataset):* For a set of QoS records $Q = QR_i$, each sample (CSPs) in the subset $Q_{Train} \subset Q$ has a complete record i.e. the trust rate is evaluated for each CSP in Q_{Train}. The remaining samples $(Q - Q_{Train})$ with the undetermined trust rate were represented as Q_{Test}. The trust rate dataset is represented as a matrix $[\mathrm{T_D}]_{mXn}$, where each CSP and corresponding QoS attributes along with its trust value is represented as rows and columns respectively.

Definition 2.1.4 *(Trust Rate Prediction Problem)*: Given a training dataset (Q_{Train}) with the QoS values and trust rate for each sample, the prediction model should provide the trust rate for each CSP in the Q_{Test} based on the experience gained from the QoS and trust values in Q_{Train}.

2.2 Probabilistic Neural Network

Probabilistic Neural Network (PNN) is a supervised, multi-layer neural network model which maps the input patterns to the output patterns in a single pass through the application of the principle of statistics (D. Specht, 1988). Unlike traditional neural network, PNN is relatively faster and has better generalization ability, since it based on both Parzen's approach to estimate probability density function (PDF) of the random variables and Bayesian strategy for decision making (D. Specht, 1990; D. F. Specht, 1990). Let us consider a

multi-class classification problem with 'c' number of classes and a training dataset $D_T = \{x_i, y_i\}_{i=1}^{n}$, where n is the total number of training samples represented by a 'm' tuple vector ('m' dimensions). According to the Bayes' decision rule, a sample (x) belongs to the class C_i, if $p(C_i)p(X|C_i) \geq p(C_j)p(X|C_j), \forall i = \{1,2,\dots,c\}$ where, $p(C_i)$ is probability of the sample (x) belongs to the class C_i, $p(X|C_i)$ is the class conditional probability density function of x in the class C_i. Then, $f_i(X) = p(C_i)p(X|C_i), f_i(X)$ is the Baye's decision function and $f_i(X) > f_k(X)\ for\ k \neq i$, is the Baye's decision rule. Further, the Baye's decision rule can be rewritten for PNN as $h_i g_i(X) > h_k g_k(X)$ for $k \neq i$, where $h_i g_i(X) = f_i(X), h_i$ is the probability of priori occurrence, g_i is the probability density function.

As shown in Figure 1, PNN consists of four layers, namely (i) ***Input layer*** with 'm' neurons for m −dimensional input feature vector, (ii) ***Pattern layer*** with 'n' neurons for n number of training samples, (iii) ***Summation layer*** with 'c' neurons that corresponds to the number of classes, and (iv) ***Output layer*** with one neuron for decision making. In this approach we have utilized the standardized Gaussian kernel function (Eqn. (1)) as PDF to compute the output of the pattern layer (D. F. Specht, 1990).

$$\vartheta_{ij}(\mathrm{x}) = \frac{1}{\sigma^2 * (2\pi)^{d/2}} [\exp[\frac{(\mathrm{x}-\mathrm{x_{ij}}) * (\mathrm{x}-\mathrm{x_{ij}})^{\mathrm{T}}}{2\sigma^2}]] \quad (1)$$

Where,

- $x = [x_1, x_2, \dots x_m]$ is the test sample with m dimension.
- x_{ij} is the j^{th} training sample of the i^{th} class
- σ=[0,1] is the smoothening factor

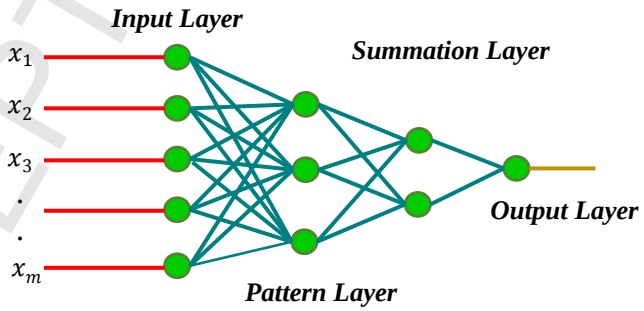


Figure 1: Probabilistic Neural Network (PNN)

Hence, the conditional probability of x belonging to the i^{th} class, which is the output of the summation layer is computed using Eqn. (2).

$$P_i(x) = \frac{1}{N_i} (\sum_{j=1}^{N} \vartheta_{ij}(\mathrm{x})) \quad (2)$$

where, N_i is the number of samples in the i^{th} class.

Finally, the class of an unknown pattern x is obtained in the output layer using Eqn. (3).

$$\text{Class(x)} = \arg\max_i \left(P_i(x)\right) \tag{3}$$

2.3 Hypergraph

Over a decade, graphical representations have been widely used to model binary relationships among the objects in various complex real-world problems. Formally, a graph $G = (v, e)$ is an ordered pair of finite set of vertices (v) and the interrelationships among the objects (e) (Gauthama Raman, Kirthivasan, & Shankar Sriram, 2017). However, in a real-world scenario, it is difficult to model multiple relationships (n-ary) among the objects using traditional graph theory. Hypergraph, a mathematical framework and a generalization of conventional graph theory expresses higher order relations among the objects in a more elegant manner (topology and geometric metrics) (Berge & Minieka, 1973). As a generic data representation framework, hypergraph along with its exciting properties (Helly, Vertex Linearity, Minimal transversal, hyper clique etc.) make the researchers in various domains to realize its benefits in terms of minimal time complexity (Gauthama Raman, Kirthivasan, et al., 2017; Gauthama Raman, Somu, Kirthivasan, Liscano, & Shankar Sriram, 2017; M. Raman, Kannan, & Pal, 2016; M. R. G. Raman et al., 2017; Somu, Kirthivasan, & Shankar, 2017; Somu, Kirthivasan, & Sriram, 2017; Somu, Raman, Kirthivasan, & Sriram, 2016). This section discusses some basic definitions of the hypergraph and its properties for dimensionality reduction.

Definition 2.3.1: (Hypergraph) A hypergraph is defined as $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \dots, v_m\}$ is the finite set of vertices and $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$ represent the hyperedges such that $e_i \subseteq \mathcal{V}, \forall i = \{1,2,\dots,n\}$ (Berge & Minieka, 1973). Figure 2 represents the hypergraph structure with 10 vertices ($\mathcal{V} = \{v_1, v_2, \dots, v_{10}\}$) and 4 hyperedges ($\mathcal{E} = \{e_1, e_2, e_3, e_4\}$); $e_1 = \{v_1, v_6, v_9, v_{10}\}$; $e_2 = \{v_6, v_1, v_3, v_2\}$; $e_3 = \{v_{10}, v_5, v_2, v_6\}$; $e_4 = \{v_8, v_4, v_7, v_5, v_{10}\}$

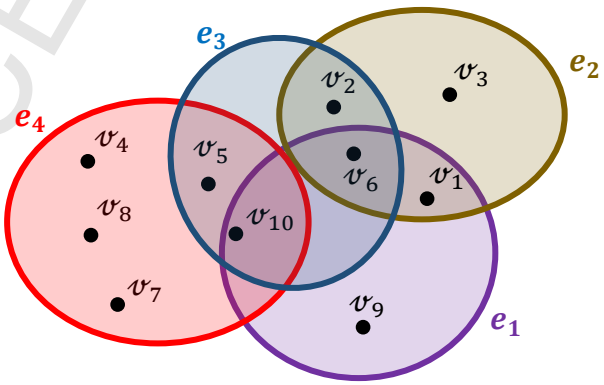


Figure 2: Hypergraph Structure

Definition 2.3.2: (Cardinality) Let $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ be a hypergraph, for each vertex $a \in \mathcal{V}$, the star in $\mathcal{H}$ denoted by $\mathcal{H}(a)$, corresponds to the set of hyperedges that contains a. The degree of a is the cardinality of $\mathcal{H}(a)$ (Berge & Minieka, 1973). Figure 3 depicts the hypergraph representation model with 7 vertices ($\mathcal{V} = \{v_1, v_2, \ldots, v_7\}$), and 5 hyperedges ($\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$); the star centered on vertex v_7 is $\mathcal{H}(v_7) = \{e_3, e_4\}$ with the degree 2.

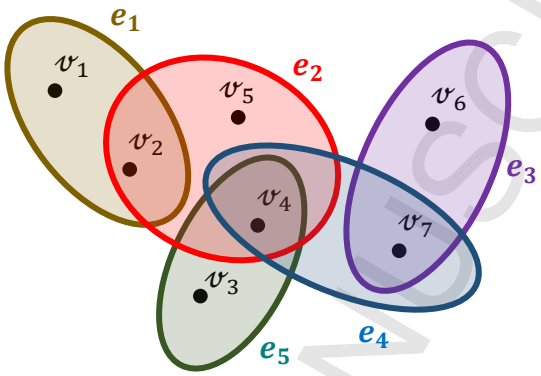


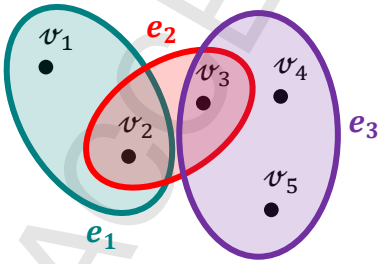
Figure 3: Hypergraph representation model

Definition 2.3.3: (Weighted Hypergraph) A weighted hypergraph is represented as $\mathcal{H}_{\boldsymbol{w}} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{w}\}$ which consists of positive number $\boldsymbol{w}(e)$ associated with each hyperedge $e \in \mathcal{E}$ known as weight of the hyperedge. A hyperedge e is said to be incident with vertex v, when v belongs to e. In general, any hypergraph ($\mathcal{H}$) (Figure 4(a)) can be represented as an incident matrix $I_{\mathcal{H}} = |\mathcal{V}|\, x |\mathcal{E}|$ using Eqn. (4) (Figure 4(b)) (Berge & Minieka, 1973).

$$\mathcal{H}(v, e) = \begin{cases} 1, if\ v \in e \\ 0, otherwise \end{cases} \tag{4}$$

The degree of each vertex is computed based on $I_{\mathcal{H}}$ (Eqn. (5)).

$$D_v = \sum_{\{e \in \mathcal{E}\}} \boldsymbol{w}(e)\mathcal{H}(v, e) \tag{5}$$



$$\mathcal{I}_{\mathcal{H}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 4: (a) Hypergraph; (b) Corresponding Incident matrix ($\mathcal{I}_{\mathcal{H}}$)

3. Proposed Hypergraph based Robust Heteroscedastic PNN for Trust prediction

In this section, we discuss the proposed Hypergraph-based Robust Heteroscedastic PNN (HC-RHRPNN) for trust prediction in cloud environments. Initially, we discuss the overall working of HC-RHRPNN followed by the description of two major modules, namely (i) Pruning and (ii) Prediction.

3.1 HC-RHRPNN: Hypergraph based Robust Heteroscedastic PNN

As in Figure 5, HC-RHRPNN comprises of three modules, namely (i) Preprocessing, (ii) Pruning, and (iii) Prediction.

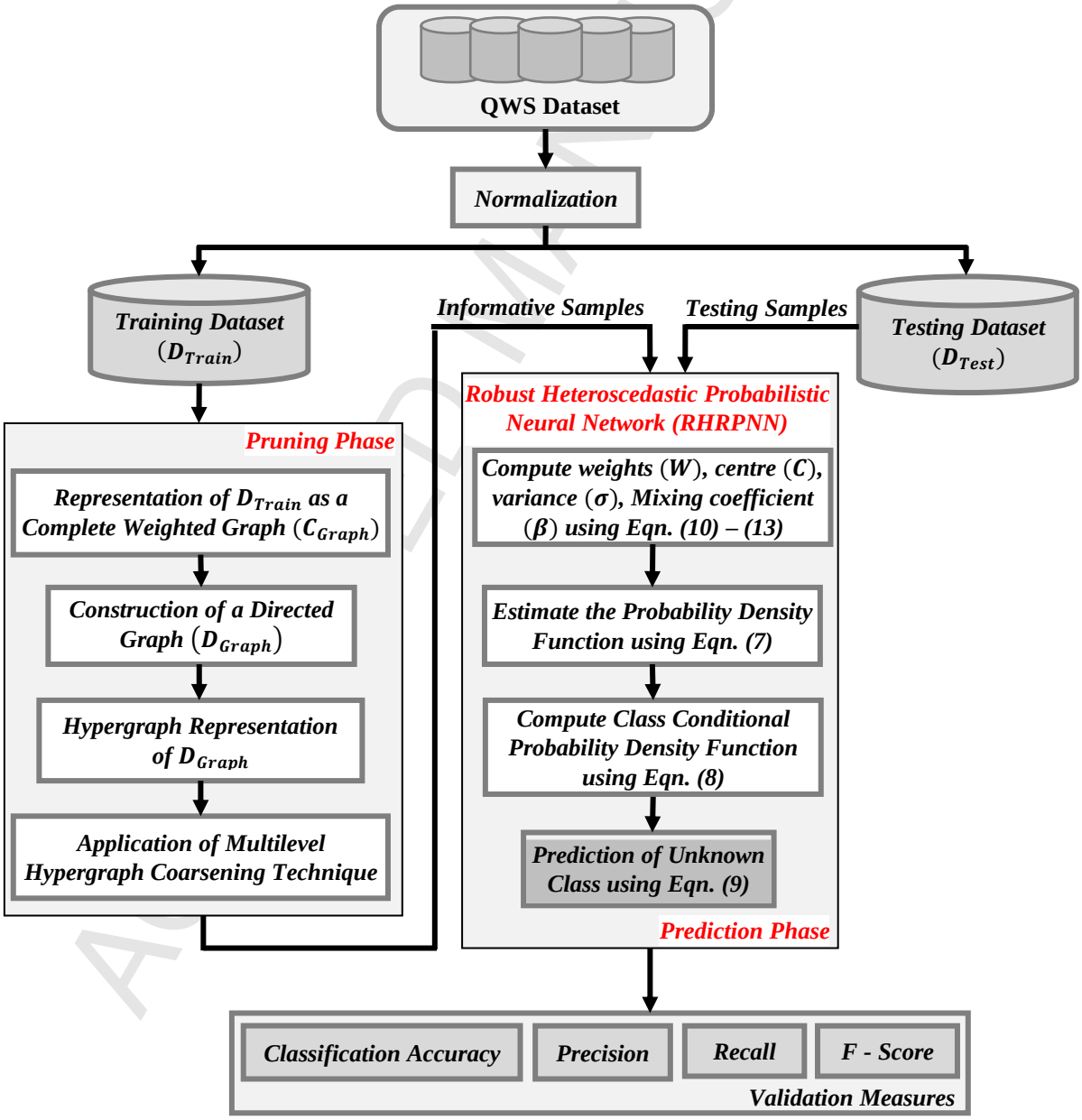


Figure 5: Hypergraph Coarsening based Heteroscedastic Probabilistic Neural Network (HC-RHRPNN)

In the first module (preprocessing), various preprocessing techniques like (i) data normalization, (ii) analysis of input and its corresponding output vectors, and (iii) designing the topological structure of HRPNN i.e. identification of number of units (neurons) in input, pattern, and summation layers, etc. were carried out. The detailed explanation of each preprocessing technique is given in section 4. Further, the given input dataset ($\text{Input}_{\text{Data}}$) with S_n samples is divided into training (D_{Train}) and testing dataset (D_{Test}) in the ratio of 80:20 using random sampling without replacement technique. The second module (pruning) is to minimize the size of the training dataset (D_{Train}) through the identification of informative samples thereby the computational overload of the subsequent prediction module is minimized.

Hence, we attempt to construct a Hypergraph structure through inducing $n-$ary relationship among the samples, for the identification of minimal set of informative samples through the recursive application of hypergraph coarsening technique. Finally, in the prediction module we train the RHRPNN, using the training patterns identified from the pruning module to compute the optimal values of kernel parameters and weights for the effective prediction of trust values of cloud service providers.

3.1.1 Pruning module

The aim of the pruning module (dimensionality reduction) is to identify the most significant i.e. informative samples in the training data, thereby improving the performance of the learning model by ensuring reliability and correctness of the input dataset. Mathematically, if training dataset $D_{Train} = \{x_1, x_2 .. x_n\} \in \mathbb{R}^{n\ x\ f_n}$ consist n number of samples represented by f_n feature vector, the dimensionality reduction process will identify a smaller dataset ($D_{Prune} \in \mathbb{R}^{r\ x\ f_n}$) which is a subset of D_{Train} such that $r \ll n$ with minimal information loss. This module consists of two major phases, namely (i) Construction of hypergraph using D_{Train} (ii) Application of hypergraph coarsening. In the initial phase, we construct a complete weighted graph (C_{Graph}) from the given D_{Train} where its vertices corresponds to the samples and the edges between the samples (vertices) are weighted using Euclidean distance metric. The Euclidean distance (E_d) metric for given sample (x, y), represented by a m tuple feature vector is computed using Eqn. (6).

$$E_d(x, y) = ((x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_m - y_m)^2)^{1/2} \quad (6)$$

For example, as illustrated in Table 2, we have considered training dataset (D_{Train}) with 10 samples ($D_{Train} = \{x_1, x_2, \ldots, x_{10}\}$) which is represented by 10 tuple feature vector. Table 3 presents the normalized form of D_{Train}. A weighted matrix [**W**] is computed using the normalized form of D_{Train} (Table 4) through the

application of Euclidean distance metric among the samples. A complete weighted graph (C_{Graph}) representation of [**W**] is reported in Figure 6. From definition 2.2.1, it is evident that hypergraph exhibits n-ary relations among the variables. Hence, it is necessary to induce multiple relationships among the samples for the formulation of hypergraph. Therefore, a directed graph (D_{Graph}) is obtained from C_{Graph} by considering the k −nearest neighbour of each vertex (Figure 7) (Kang, 2011). For every vertex $x_i, i = (1,2,\dots,n)$ we select a similar vertex or a nearest vertex based on the weighted edges. For example, the nearest neighbors of vertex x_1 in C_{Graph} (Figure 6) are x_2 and x_3 which is evident from Table 4 ($\boldsymbol{d_{(x_1,x_2)} = 0.09 < d_{(x_1,x_3)} = 0.14 <}$ $d_{(x_1,x_4)} = 0.15, \ll d_{(x_1,x_5)} = 0.28 = d_{(x_1,x_7)} = 0.28 < d_{(x_1,x_6)} = 0.33 < d_{(x_1,x_8)} = 0.39 \ll d_{(x_1,x_9)} = 0.58 <$ $d_{(x_1,x_{10})} = 0.64$). Each vertex (samples) in D_{Graph} (Figure 7(a)) consists of multiple directed edges from other vertices implying an n −ary relationship among the vertices, therefore D_{Graph} can be effectively represented as a hypergraph ($\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$) as shown in Figure 8. In simpler terms, the construction of hypergraph (Figure 8) from the D_{Graph} (Figure 7(a)) and its corresponding vertex-edge representation (Figure 7(b)) is based on the multiple directed edges i.e., multiple relations among the vertices $\big((x_1, x_3, x_4, x_2), (x_9, x_{10}, x_8), (x_7, x_6, x_5), (x_8, x_7, x_6)\big)$.

Table 2: Sample Dataset

Samples	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
x_1	45	83	27.2	50	97.4	89	91	43	58	100
x_2	71.75	100	14.6	88	85.5	78	80	64.42	86	93
x_3	125.5	100	16.4	80	89.2	78	84	125	93	88
x_4	126.77	100	3.2	86	81.8	78	80	119.44	95	81
x_5	261	100	1.8	71	58.1	78	80	229.5	94	71
x_6	307.5	100	0.7	71	58.4	78	77	301	94	69
x_7	176	56	8	33	56.7	89	91	172	7	55
x_8	370.71	60	4.2	51	64.1	78	77	221.98	7	52
x_9	1126.25	67	3.5	33	59.4	78	72	1107	1	45
x_{10}	1680.16	16	2	7	12.5	89	72	1667.22	11	30

Table 3: Normalized Sample Dataset

Samples	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
x_1	0.07	0.12	0.04	0.07	0.14	0.13	0.13	0.06	0.08	0.15
x_2	0.09	0.13	0.02	0.12	0.11	0.10	0.11	0.08	0.11	0.12
x_3	0.14	0.11	0.02	0.09	0.10	0.09	0.10	0.14	0.11	0.10
x_4	0.15	0.12	0.00	0.10	0.10	0.09	0.09	0.14	0.11	0.10
x_5	0.25	0.10	0.00	0.07	0.06	0.07	0.08	0.22	0.09	0.07
x_6	0.27	0.09	0.00	0.06	0.05	0.07	0.07	0.26	0.08	0.06
x_7	0.24	0.08	0.01	0.04	0.08	0.12	0.12	0.23	0.01	0.07
x_8	0.38	0.06	0.00	0.05	0.07	0.08	0.08	0.23	0.01	0.05
x_9	0.43	0.03	0.00	0.01	0.02	0.03	0.03	0.43	0.00	0.02
x_{10}	0.47	0.00	0.00	0.00	0.00	0.02	0.02	0.46	0.00	0.01

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	0	0.09	0.14	0.15	0.28	0.33	0.28	0.39	0.58	0.64
x_2	0.09	0	0.09	0.09	0.23	0.28	0.25	0.36	0.55	0.60
x_3	0.14	0.09	0	0.02	0.15	0.19	0.18	0.28	0.46	0.52
x_4	0.15	0.09	0.02	0.00	0.15	0.19	0.18	0.28	0.46	0.52
x_5	0.28	0.23	0.15	0.15	0.00	0.05	0.11	0.16	0.32	0.37
x_6	0.33	0.28	0.19	0.19	0.05	0.00	0.12	0.14	0.27	0.33
x_7	0.28	0.25	0.18	0.18	0.11	0.12	0.00	0.15	0.32	0.38
x_8	0.39	0.36	0.28	0.28	0.16	0.14	0.15	0.00	0.23	0.29
x_9	0.58	0.55	0.46	0.46	0.32	0.27	0.32	0.23	0.00	0.06
x_{10}	0.64	0.60	0.52	0.52	0.37	0.33	0.38	0.29	0.06	0.00

Table 4: Weighted Matrix

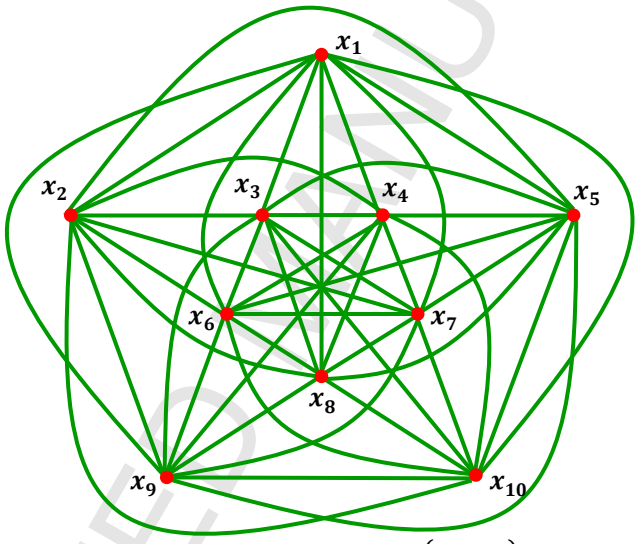
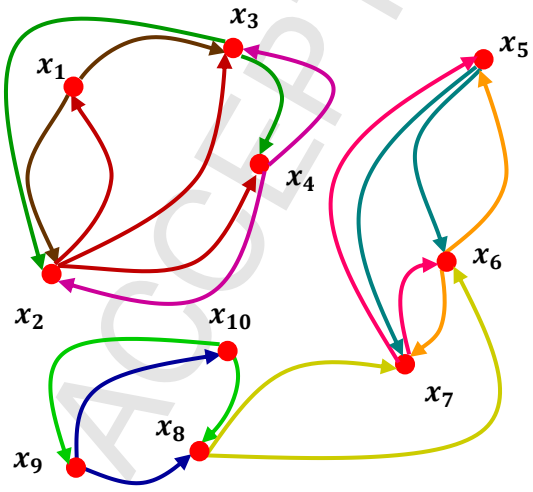


Figure 6: Complete Graph (C_{Graph})



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	0	1	1	0	0	0	0	0	0	0
x_2	1	0	1	1	0	0	0	0	0	0
x_3	0	1	0	1	0	0	0	0	0	0
x_4	0	1	1	0	0	0	0	0	0	0
x_5	0	0	0	0	0	1	1	0	0	0
x_6	0	0	0	0	1	0	1	0	0	0
x_7	0	0	0	0	1	1	0	0	0	0
x_8	0	0	0	0	0	1	1	0	0	0
x_9	0	0	0	0	0	0	0	1	0	1
x_{10}	0	0	0	0	0	0	0	1	1	0

Figure 7: Directed Graph (D_{Graph}) and its Vertex-Edge Representation

To further enhance the understandability on the construction of hypergraph let us consider the vertices x_1, x_3, x_4, and x_2 in Figure 7(a). Among these vertices, since x_3, x_4, and x_2 have multiple directed edges i.e., in degree and out degree of the vertices ($x_2^{ID} = 3;\ x_3^{ID} = 3; x_2^{ID} = 2; x_2^{OD} = 3; x_3^{OD} = 2; x_4^{OD} = 2$) are greater than or equal to 2, they constitute to the hyperedge (e_3) in the hypergraph ($\mathcal{H}$) (Table 5). The vertex x_1 does not constitute the hyperedge (e_3) since its in degree is 1 ($x_1^{ID} = 1$) and out degree is 2 ($x_1^{OD} = 2$). In a more similar way, the entire hypergraph structure given in Figure 8 was constructed based on the multiple directed edges among the vertices in the D_{Graph}.

Table 5: In Degree and Out Degree of D_{Graph}

Vertex	Indegree (x_i^{ID})	Outdegree (x_i^{OD})
x_1	1	2
x_2	3	3
x_3	3	2
x_4	2	2
x_5	2	2
x_6	3	2
x_7	3	2
x_8	2	2
x_9	1	2
x_{10}	1	2

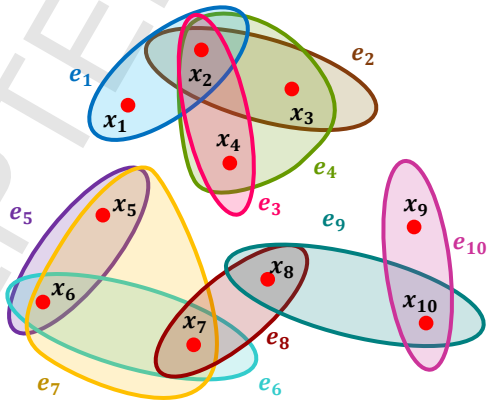


Figure 8: Hypergraph Representation of D_{Graph}

On to the subsequent phase, the recursive application of hypergraph coarsening was carried out to remove redundancies from the training dataset. During this process, we merge the pair of vertices in a recursive manner which results in a smaller hypergraph $(\dot{\mathcal{H}} = \{\dot{\mathcal{V}}, \dot{\mathcal{E}}\})$ such that $|\dot{\mathcal{V}}| \ll |\mathcal{V}|$ and $|\dot{\mathcal{E}}| \ll |\mathcal{E}|$. The vertices of

the hypergraph ($\mathcal{H}$) were merged using the maximum weight matching problem, which identifies the perfect match for each vertex ($\mathcal{V}$) that maximizes the sum of the edge weight of the vertex pairs (Çatalyürek & Aykanat, 1999). For example, consider an incident matrix ($I_{\mathcal{H}}$) representation of hypergraph ($\mathcal{H}$) according to the Definition 2.2.3 in Subsection 2.2 (Figure 4). The weight of the vertices x_1 and x_{10} is zero, since they do not have any common hyperedges. Similarly, for x_5 and x_6 edge weight is two, since they are contained in hyperedges e_5 and e_7. Throughout this process, greedy search algorithm was used to explore the vertices in a random manner and the vertices with maximum weight were merged with its unmatched neighbors. This process is repeated until all the vertices in the hypergraph were matched (Haw-ren Fang & Yousef Saad, 2011). From the given illustration, it is clear that the samples x_1 and x_9 cannot be merged with other vertices, since they have least interaction with other vertices which proves that they are redundant. Finally, the coarse vertex set consists of matched vertex pairs that correspond to the optimal number of informative samples ($\mathcal{V}_{Coarse} = \{x_2, x_3, x_5, x_6, x_7, x_{10}\}$).

Algorithm 1: Hypergraph based Robust Heteroscedastic Probabilistic Neural Network (HG-RHRPNN)

Input
$Input_{Data}, S_n, f_n, c$
Output
C_{Test}

$\boldsymbol{HG - RHPNN}()$
Begin
**** Preprocessing Phase ****
Generate Training (D_{Train}), Testing (D_{Test}) dataset using $Input_{Data}$
**** Pruning Phase ****
Construct a complete weighted graph (C_{Graph}) using D_{Train} where samples as vertices and edges weighted using Eqn. (9)
Generate a Directed graph (D_{Graph}) by considering the closest neighbours of each vertex in C_{Graph}
Represent the D_{Graph} as hypergraph ($\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$) and compute the incident matrix ($I_{\mathcal{H}}$) using definition for $\mathcal{H}$
// Multilevel hypergraph coarsening based Dimensionality reduction //
$D_{Prune} \leftarrow \{\}$
While ($S \neq \{\}$) **begin**
  Randomly select a vertex $j \in S$
  $S \leftarrow S - \{j\}$
  for each $g \leftarrow 1$ to n **begin**
    $Q[g] \leftarrow 0$
  end
  for each i such that $I_{\mathcal{H}}[i, j] \neq 0$ **begin**
    for each g such that $I_{\mathcal{H}}[i, g] \neq 0$ **begin**
      $Q[g] \leftarrow Q[g] + 1$
    end for
  end for
  $i \leftarrow \arg\max\{Q[g]\} : g \in S$
  if ($Q[g] \neq 0$) **begin**

Add i^{th} vertex with its unmatched nearest neighbour j^{th} vertex in D_{Prune}
$S \leftarrow S - \{i\}$
end
end while
return D_{Prune}

**** Prediction Phase ****

for each class c **begin**
  for each sample in c **begin**
    Estimate the weights (W), centre (C), variance (σ), Mixing coefficient (β) using Eqn. (10-18)
  end
end

// Probability Density Estimation //

for each $i \leftarrow 1$ to c **begin**
  for each $j \leftarrow 1$ to n **begin**
    $\vartheta_{ij}(\mathrm{x}) = \frac{1}{(2\pi\sigma_{i,j})^{d/2}} \exp(\frac{-||x-C_{i,j}||^2}{2{\sigma_{i,j}}^2})$
  end
end

// Class Conditional Probability Density Estimation //

for each $i \leftarrow 1$ to c **begin**
  for each $j \leftarrow 1$ to n **begin**
    $P_j(x) = \sum_{i=1}^{R_j} \beta_{ij}\, \vartheta_{ij}(\mathrm{x})$
  end
end

// Class prediction //

for each $j \leftarrow 1$ to c **begin**
  $C_{Test} \leftarrow \arg\max_j(\alpha_j P_j(x))$
end

The computational complexity of the proposed hypergraph-based dimensionality reduction technique was found to be $O(n)$, since the main loop (Line 7-23) in the Algorithm 1 runs for n times (worst case). Since, the complexity incurred during the construction of hypergraph is negligible when compared to the complexity of the hypergraph coarsening, we have not considered while deriving the overall complexity of the hypergraph-based dimensionality reduction technique. Recently in 2017, (Bostani & Sheikhan, 2017) proposed a similar work where they have utilized social network concept for the identification of informative samples from a directed graph with the complexity of $O(nm + n^3\, logn)$ where n and m are the number of samples and edges respectively. It is obvious that hypergraph based pruning module has least complexity ($O(n) \lll O(nm + n^3\, logn)$) than the existing one. In addition to that, since the coarsening process operates over the sparse matrix (maximum elements are zero) the computational complexity of hypergraph based dimensionality reduction technique will be minimum compared with existing traditional dimensional reduction technique like principle component analysis (PCA), Independent component analysis (ICA) (Çatalyürek & Aykanat, 1999) etc.

3.1.2 Prediction module

The performance of any learning model can be assessed based on its degree of accuracy in identifying similar data from a collection of unknown input patterns. As a learning model, Artificial Neural Network (ANN) plays a vital role in many classification and prediction problems, since it has the ability to learn from the training dataset and a better generalization ability (M. R. G. Raman et al., 2017). In order to enhance its prediction accuracy, stability etc. various research works were carried out towards the design of its variants like Radial Basis Neural Network (RBNN), Adaptive Resonance Theory Network (ARTN), etc. One such variant is Probabilistic Neural Network (PNN), which is based on the concept of competitive learning i.e., "winner takes all attitude" (D. F. Specht, 1990). It combines the estimated conditional probabilities using the non – parametric estimator (Parzen window) to obtain the probability distribution function (PDF) for predicting the class of the unknown sample. Generally, PNN does not possess any feedback path like other traditional neural network architectures, rather it integrates kernel based estimator and radial basis function network for faster learning.

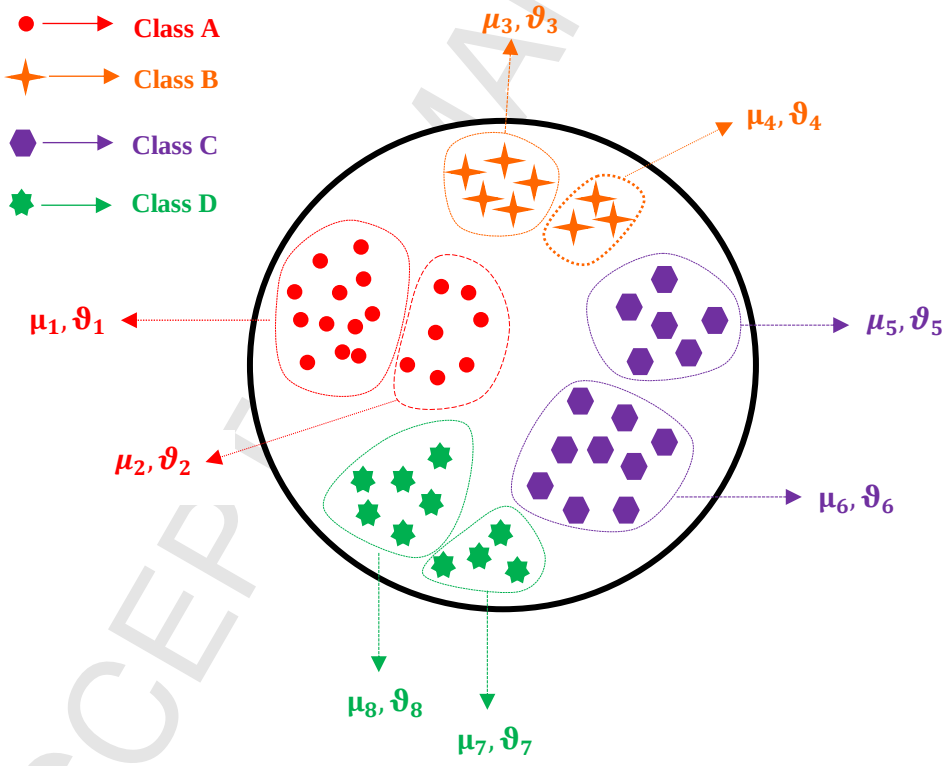


Figure 9: Variable Center and Variance in RHRPNN

The basic version of PNN utilizes all the samples in the dataset for the estimation of center or mean vector of the Gaussian kernel function which resulted in the notable increase in its classification ability. However, the detrimental effect of large number of training samples resulted in "Over training", therefore it is

desirable to utilize minimal number of kernel functions without comprising the classification accuracy. In the year 1994, Streit et al. employed Expectation Maximization (EM) algorithm to estimate the mixture of kernel functions i.e., training the two variants of PNN, namely (i) Homoscedastic PNN - Different centres with common variance and (ii) Heteroscedastic PNN (HRPNN) - Difference centres with difference variance (Figure 9).

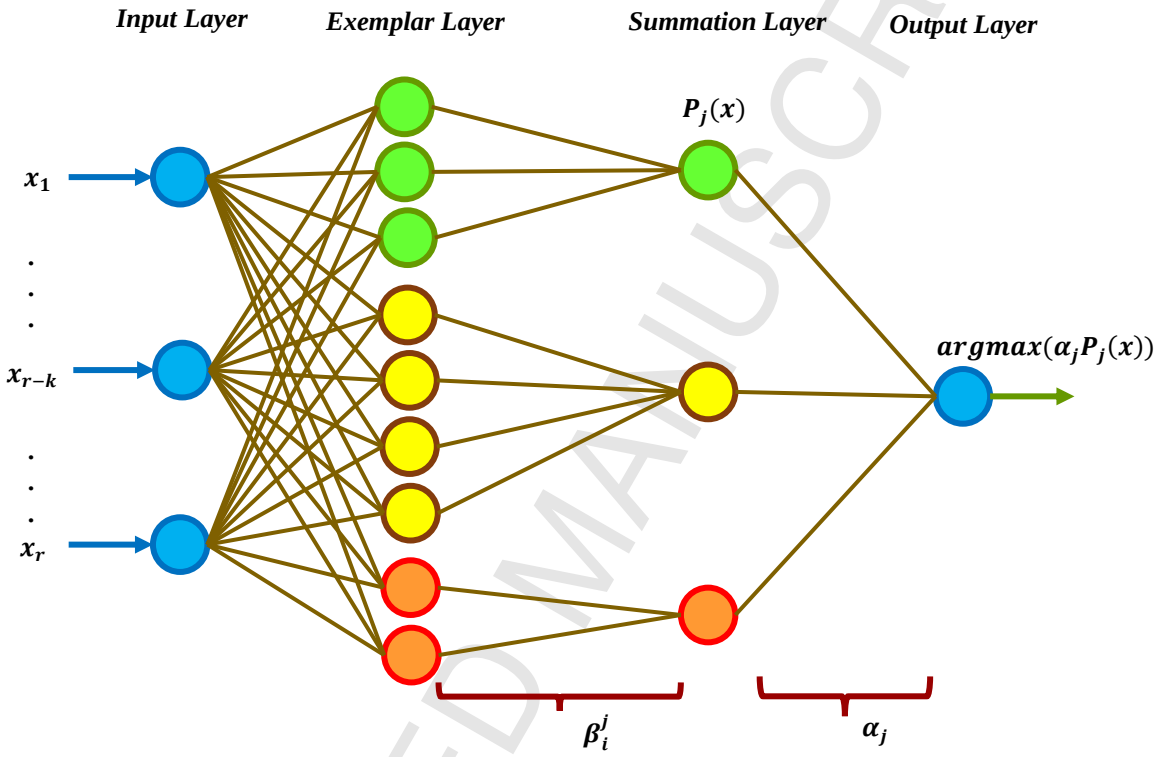


Figure 10: HRPNN Architecture

Fundamentally, the architecture of HRPNN is similar to the basic version of PNN however, the major differences can be observed in the *exemplar layer* and the *output layer*. The four-layered feed forward HRPNN is given in Figure 10. The first layer (Input layer) distributes the received input patterns to the second layer (Pattern layer) which consists of *'n'* number of neurons divided into *'c'* groups (number of classes). The Gaussian kernel function of the i^{th} node of the j^{th} class is defined in Eqn. (7) (Venkatesh & Gopal, 2011).

$$\vartheta_{ij}(\mathrm{x}) = \frac{1}{(2\pi\sigma_{i,j})^{d/2}} \exp\left(\frac{-||x-c_{i,j}||^2}{2\sigma_{i,j}{}^2}\right) \tag{7}$$

where, C and σ are the mean vector and variance respectively.

Similarly, the third layer consists of '*c*' nodes to estimate the class conditional probability density function (Eqn. (8)).

$$P_j(x) = \sum_{i=1}^{n_j} \beta_{ij}\, \vartheta_{ij}(\mathrm{x}) \,, 1 \le j \le c \tag{8}$$

such that $\sum_{i=1}^{c_j} \beta_{ij} = 1, 1 \le j \le c$ where, c_j is the number of nodes in j^{th} class.

Finally, the decision layer (Output layer) predicts the class of the unknown input pattern using Eqn. (9).

$$\mathrm{Class(x)} = \arg\left(\max_{1 \le j \le c} \left(\alpha_j P_j(x)\right)\right) \tag{9}$$

where, α_j is the class priori probability. Chattfield et al. presented robust heteroscedastic PNN (RHRPNN) which utilizes the jack knife procedure (Statistical tool) for the estimation of model parameters like center ($C_{i,j}$), variance ($\sigma_{i,j}$), and weights (β_{ij}) and to overcome the numerical difficulties in traditional EM algorithm during the estimation process while handling sparse dataset. During the estimation process, entire training patterns were divided into 'c' subsets, such that $D_{Train} = \{\{x_{i,j}\}_{i=1}^{R_j}\}_{j=1}^{c}$, where R_j is the number of samples in the j^{th} class and the impact of model parameters on each subset was observed. The two major steps involved in the training process of RHRPNN are highlighted as follows (Yang et al., 2000), where $\hat{c}_{m,i}^{(t+1)}$ and $\hat{\sigma}_{m,i}^2|^{t+1}$ are the jacknife estimates obtained from the $c_{m,i}$ and $\sigma_{m,i}$ at the t^{th} step respectively.

(i) Weights computation at step t is for $1 \le m \le M_i, 1 \le n \le R_i, 1 \le i \le c$ (Eqn. (10) and Eqn. (11))

$$W_{m,i}^{(t)}(x_{n,i}) = \frac{\beta_{m,i}(\,\vartheta_{m,i}^{(t)}(x_{n,i}))}{\sum_{s=1}^{M_i} \beta_i^s p_s^{i\,(t)}(x_n^i)} \tag{10}$$

where, $\vartheta_{m,i}^{(t)}(x_{n,i})$ is computed using Eqn. (7).

(ii) Parameter updation for $1 \le m \le M_i, 1 \le i \le c$ (Eqn. (11) – Eqn. (13))

$$\hat{c}_{m,i}^{(t+1)} = R_i c_{m,i}^{(t+1)} - \frac{R_i - 1}{R_i} \sum_{j=1}^{R_i} c_{m,i}^{(t+1)}|_{-j} \tag{11}$$

where, $c_{m,i}^{(t+1)} = \frac{\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i}) x_{n,i}}{\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i})}$; $c_{m.i}^{(t+1)}|_{-j} = \frac{\sum_{n=1;n\neq j}^{R_i} W_{m,i}^t(x_{n,i}) x_{n,i}}{\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i})}$ $\forall 1 \le j \le R_i$

Similarly,

$$\hat{\sigma}_{m,i}^2|^{t+1} = R_i \sigma_{m,i}^2|^{t+1} - \frac{R_i - 1}{R_i} \sum_{j=1}^{R_i} \sigma_{m,i}^2|_{-j}^{t+1} \tag{12}$$

Where, $\sigma_{m,i}^2|^{t+1} = \frac{\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i})\left\|x_{n,i}-\hat{c}_{m,i}^{(t)}\right\|^2}{d\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i})}$; $\sigma_{m,i}^2|_{-j}^{t+1} = \frac{\sum_{n=1,n\neq j}^{R_i} W_{m,i}^t(x_{n,i})\left\|x_{n,i}-\hat{c}_{m,i}^{(t)}\right\|^2}{d\sum_{n=1}^{R_i} W_{m,i}^t(x_{n,i})}$ $\forall 1 \leq j \leq R_i$

$$\beta_{m,i}^{t+1} = \frac{1}{R_i}\sum_{n=1}^{R_i} W_{m,i}^t\,(x_{n,i}) \tag{13}$$

4. Experimental Results and Analysis

4.1 Dataset Description

QWS, a public QoS dataset created and maintained by E. Al-Masri, and Q.H Mahmooud, University of Guelph, Canada (Al-Masri & Mahmoud, 2007). This dataset has been extensively used in various research works on QoS in service-oriented environments (web service, cloud computing, etc.). It consists of QoS records of 365 real web services collected using Web Service Crawler Engine (WSCE). Each QoS record consists of nine QWS attributes like response time, availability, throughput, and so on (Table 6). Each service was tested over a period of ten minutes for three successive days. The trust rate was evaluated using Web Service Relevancy Function (WsRF) based on quality metrics (response time, availability, throughput, successability, reliability, compliance, best practices, and latency). Each web service was classified into four levels, namely platinum (high quality), gold, silver, and bronze (low quality) based on the overall rating provided by WsRF. The four levels of service offering qualities were represented by numbers 1 to 4, respectively. The service classification (TMP_{11}) is measured based on the overall ranking given by WsRF.

Table 6: QWS Trust Measure Parameters and Units

ID	Trust Measure Parameter	Description	Units	Conditional/ Decisional Attribute
TMP_1	Response Time	Time taken to send a request and receive a response	ms	Conditional
TMP_2	Availability	Number of successful invocations/total invocations	%	Conditional
TMP_3	Throughput	Total number of invocations for a given period of time	Invokes/ s	Conditional
TMP_4	Successability	Number of response/number of request messages	%	Conditional
TMP_5	Reliability	Ratio of the number of error messages to total messages	%	Conditional
TMP_6	Compliance	The extent to which a WSDL document follows WSDL documentation	%	Conditional
TMP_7	Best Practices	The extent to which a web service follows	%	Conditional
TMP_8	Latency	Time taken for the server to process a given request	ms	Conditional
TMP_9	Documentation	Measure a documentation (i.e. description tags) in WSDL	%	Conditional

TMP_{10}	WSRF	Web service relevance function: a rank for web service quality	%	Conditional
TMP_{11}	Service classification	Levels representing service offering qualities (1 through 4)	Classifier	Decisional
TMP_{12}	Service name	Name of the web services	None	Ignored
TMP_{13}	WSDL address	Location of the web service definition language (WSDL) file on web	None	

4.2 Experimental Setup

The proposed technique (HC-RHRPNN) was implemented using Python 3.6 on an INTEL® Core™ i5 processor @ 2.40 GHz system with 8 GB RAM running Windows 7 operating system. In addition, WEKA tool and Matlab R2016b were used for validation purposes (Witten, Ian H., Eibe Frank, Mark A. Hall, 2016). The entire set of experiments were divided into three phases, namely (i) Data pre-processing, (ii) Generation of training and testing datasets, and (iii) Performance validation using various quality metrics.

4.2.1 Data Preprocessing

During the initial phase of the experiment, *data pre-processing* was used to transform the QWS public dataset into a compatible format supported by the classifiers. In this phase, data normalization was carried out to reduce the impact of the features with high value. Each feature in the sample is normalized such that all the values lie in the range of [0,1]. In this study, we have applied Min-Max normalization technique on each sample in the considered dataset (Eqn. 14).

$$f_{ab} = \frac{f_{ab} - (f_{ab})_{Min}}{(f_{ab})_{Max} - (f_{ab})_{Min}};\ \forall\ a = (1,2,\ldots,S); b = (1,2,\ldots,n) \quad (14)$$

where,

S and n - total number of samples and features in the dataset respectively

f_{Min} and f_{Max} - minimum and maximum value of the feature in a sample.

4.2.2 Generation of Training and Testing Datasets

On to the subsequent phase, the training (D_{Train}) and testing (D_{Test}) dataset were generated using *random sampling without replacement technique.* Finally, to show the predominance of HC-RHRPNN, it was compared with the traditional classifiers like BayesNet, CART, Random Forest, and SVM & few versions of neural network like PNN, BPNN, and MLPNN. The available packages in Weka tool and MatLab neural network

toolbox were used for the implementation of the existing classifiers. Further, the performance and effectiveness of HC-RHRPNN over the existing classifiers were assessed using 10 fold cross validation based on the following metrics.

(i) **Classification Accuracy** (***Acc***)**:** It represents the degree of correct predictions among all the classes. The mathematical formulation of classification accuracy is given in Eqn. (15).

$$\textbf{Accuracy } (\boldsymbol{Acc})\textbf{:}\ \frac{T_P+T_N}{T_P+T_N+F_P+F_N} \tag{15}$$

(ii) **Precision** (***Prec***)**:** Precision or Positive Predictive Value (PPV) is the measure of the exactness of the learning model (Eqn. (16)).

$$\boldsymbol{Precision\ (Prec)\ :}\ \frac{T_P}{T_P+F_P} \tag{16}$$

(iii) **Recall** (***RC***)**:** Recall or True Positive Rate (TPR) is the measure of the completeness of the learning model (Eqn. (17)).

$$\boldsymbol{Recall\ (RC):}\ \frac{T_P}{T_P+F_N} \tag{17}$$

(iv) **F-Score** (***FS***)**:** F-Score or F-Measure is the harmonic mean or the balance between the precision ($Prec$) and recall (RC) metric (Eqn. (18)).

$$\boldsymbol{F\text{-}Score\ (FS)}=\frac{2*Prec*RC}{Prec+RC} \tag{18}$$

where, T_P, F_P, T_N, and F_N are true positive, false positive, true negative and false negative respectively. To gain better understanding, for a binary classification problem (Class A and Class B), T_P, F_P, T_N, and F_N can be defined as follows,

True Positive ($\boldsymbol{T_P}$)**:** Measure of samples correctly classified class A as class A

True Negative ($\boldsymbol{T_N}$)**:** Measure of samples correctly classified class B as class B

False Positive ($\boldsymbol{F_P}$)**:** Measure of samples misclassified class A as class B

False Negative ($\boldsymbol{F_N}$)**:** Measure of samples misclassified class B as class A

4.3 Results and Discussions

In this section, we briefly discuss various experimental and comparative analysis carried out to demonstrate the performance of HC-RHRPNN over the classical classification algorithms and neural network variants in terms of various quality metrics discussed in Section 4. The major motive behind the design of an efficient trust prediction model is to achieve high prediction accuracy with minimal time complexity. The major objective of this work is to improve the performance of HRPNN over the several neural network variants for trust prediction problem. Initially, we prove the significance of pruning model by carrying out the experiments under two scenarios, namely (i) ***Without pruning:*** classifiers trained with all samples and (ii) ***With pruning:*** classifiers trained with the informative samples obtained from the pruning module (hypergraph coarsening), in terms of classification accuracy for different ratio of training dataset. As reported in Table 7, it is clear that HRPNN outperforms the existing classifiers in terms of classification accuracy under both the scenarios.

Further, except few exceptions, the performance of the existing classifiers and HRPNN increases with the increase in the ratio of the training dataset. An important point to note is that the performance of the classifiers increases when trained with the informative samples obtained from the proposed hypergraph coarsening based pruning module. For example, the classification accuracy of SVM, a well-known classifier increases from 65.73%, 72.42%, 70.91%, and 70.87% (without pruning) to 72.17%, 75.33%, 74.21%, and 83.63% (with pruning) respectively for different ratios of D_{Train}. Another interesting fact is that the overwhelming performance of HRPNN under both scenarios for all ratios of D_{Train} was due to the use of multiple Gaussian kernel functions and their variance based on the characteristics of the dataset.

Table 7: Performance Validation - Classification Accuracy for Different Ratio of Training Dataset

S.No	Classifiers	Without Pruning				With Pruning			
		D_{Train} Ratio				D_{Train} Ratio			
		50%	60%	70%	80%	50%	60%	70%	80%
1	Bayes Net	55.30	55.07	54.28	61.02	57.83	58.62	58.72	72.53
2	SVM	65.73	72.42	70.91	70.87	72.17	75.33	74.21	83.63
3	CART	57.86	57.03	67.27	46.31	62.93	68.14	70.14	64.15
4	J48	40.89	47.13	50.02	57.82	43.87	51.86	52.50	68.49
5	C4.5	65.96	71.59	71.35	62.94	74.61	75.22	74.86	73.82
6	Random Forest	73.47	76.12	80.10	58.62	81.75	83.17	82.77	61.87
7	BPNN	76.10	81.60	82.07	72.18	83.83	84.62	85.60	81.76
8	MLPNN	78.33	82.23	81.02	78.92	86.11	85.91	86.71	85.62
9	PNN	75.40	82.07	82.97	62.87	84.82	85.61	85.10	86.11
10	**RHRPNN**	**78.83**	**83.05**	**84.12**	**80.21**	**88.76**	**87.12**	**87.62**	**94.63**

The significant improvement in the classification accuracy of the learning models reveals the impact of pruning module on the performance of the classifiers. Thereby, for further experimentations, we have trained the classifiers with the informative samples identified by our proposed hypergraph-based pruning module. In order to minimize the complexity of the construction of a complete graph (initial phase of the pruning module), we have identified the significant features using Rough Set – Hypergraph based feature selection Technique (RSHT), our previous research work on feature selection for the identification of optimal trust measure parameters (Somu, Kirthivasan, & Sriram, 2017).

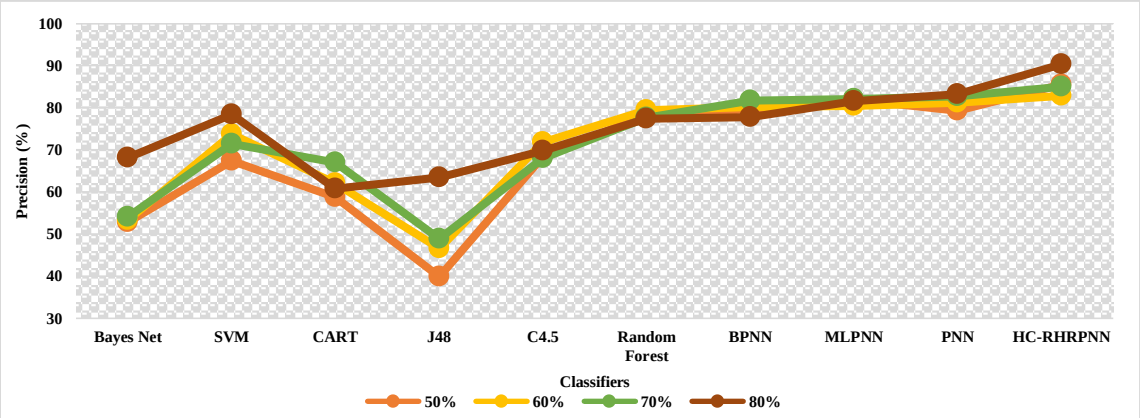


Figure 11: Performance Validation – Precision

On to the subsequent phase, the performance of HC-RHRPNN was further analyzed in terms of precision, recall, and F-Score for different ratios of D_{Train}. As shown in Figure 11-13, the performance of HC-RHRPNN was found to be predominantly higher than the traditional classification algorithms like Bayes Net, SVM, CART, J48, C4.5, and Random Forest. For example, under different ratio of D_{Train}, the precision value of Bayes Net, SVM, CART, J48, C4.5, and Random Forest is less than 80%, whereas the precision value of HC-RHRPNN lies in the range of 82%-90% (approx.). Similarly, in the case of recall and F-Score, the overall performance of the traditional classification algorithms did not exceed 80%, however HC-RHRPNN achieves a maximum of 92.33% (recall) and 91.29 (F-Score).

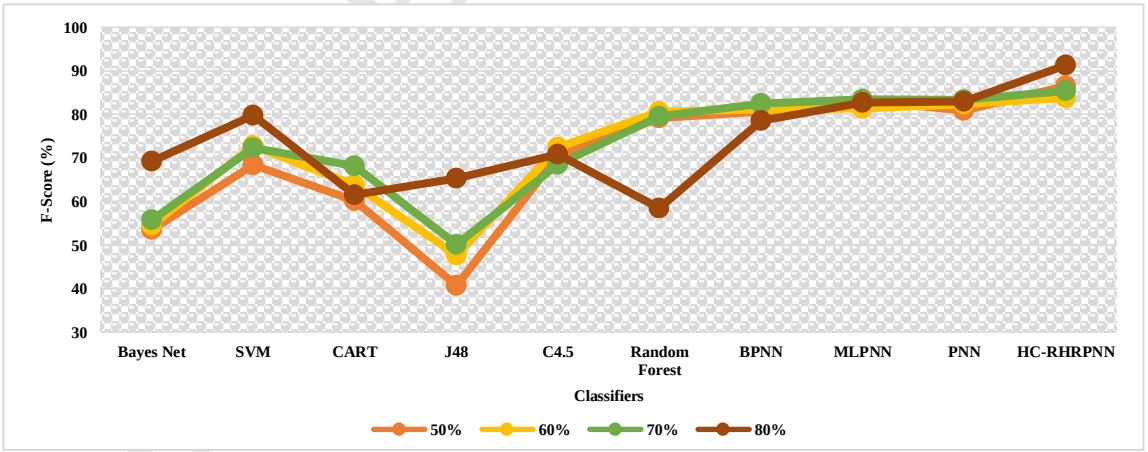


Figure 12: Performance Validation – Recall

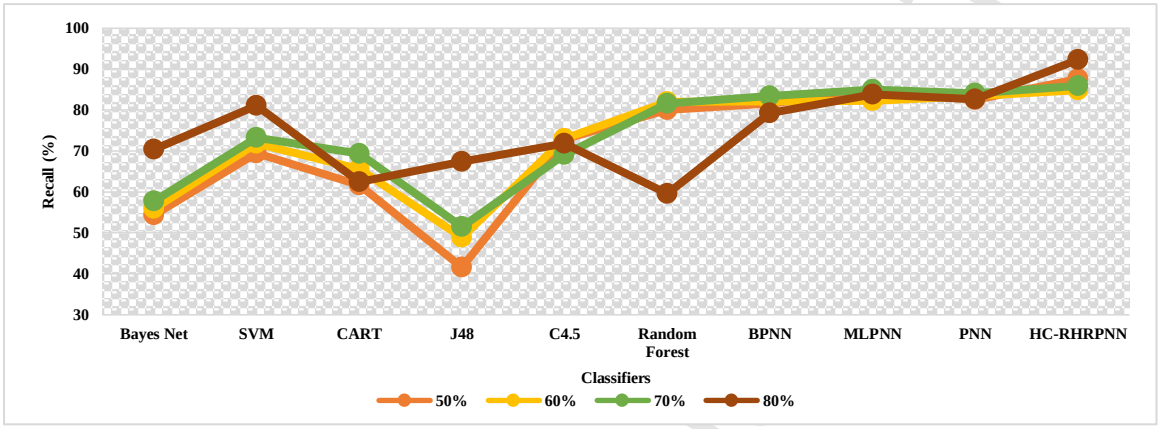


Figure 13: Performance Validation – F-Score

In addition, from Table 7 and Figure (11-13), it is evident that the difference in the performance of HRPNN and existing classifiers (Bayes Net, SVM, CART, J48, C4.5, Random Forest; Classification accuracy: +45%, Precision, recall and F-Score: +51%) was found to be larger than PNN, BPNN, and MLPNN (Classification accuracy: +2%, Precision, recall and F-Score: +6%) i.e. HRPNN has similar performance to NN variants. Hence, we have performed a detailed analysis by plotting distributions of various quality metrics for different ratio of training datasets to prove the effectiveness of HC-RHRPNN over the variants of neural networks.

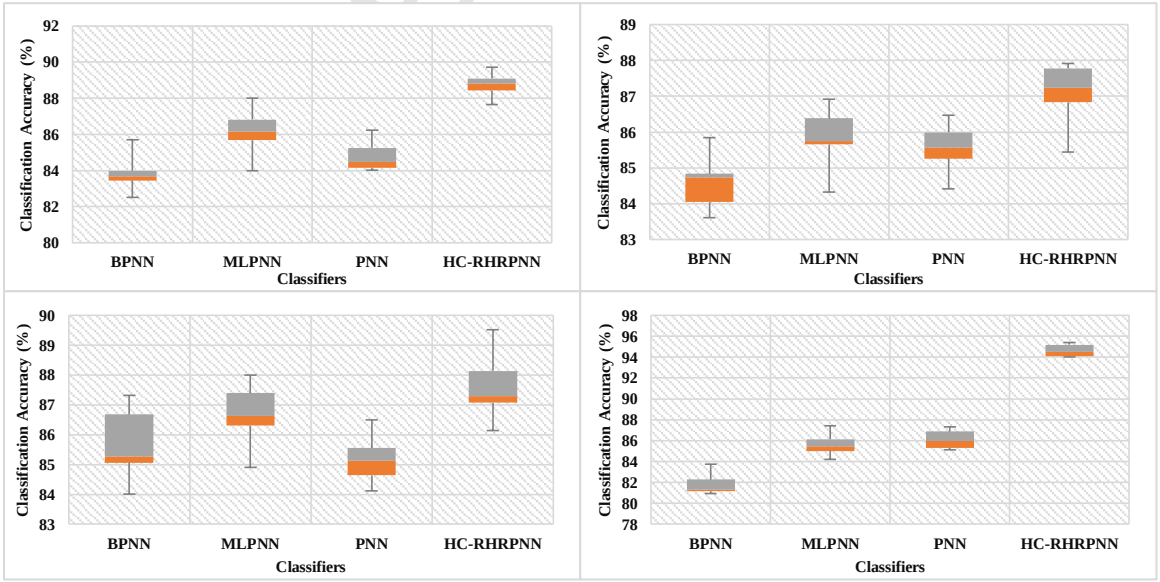


Figure 14: Classification Accuracy Distribution in Different Ratio of Training Datasets (a) Training dataset ratio=50%; (b) Training dataset ratio=60%; (c) Training dataset ratio=70%; (d) Training dataset ratio=80%

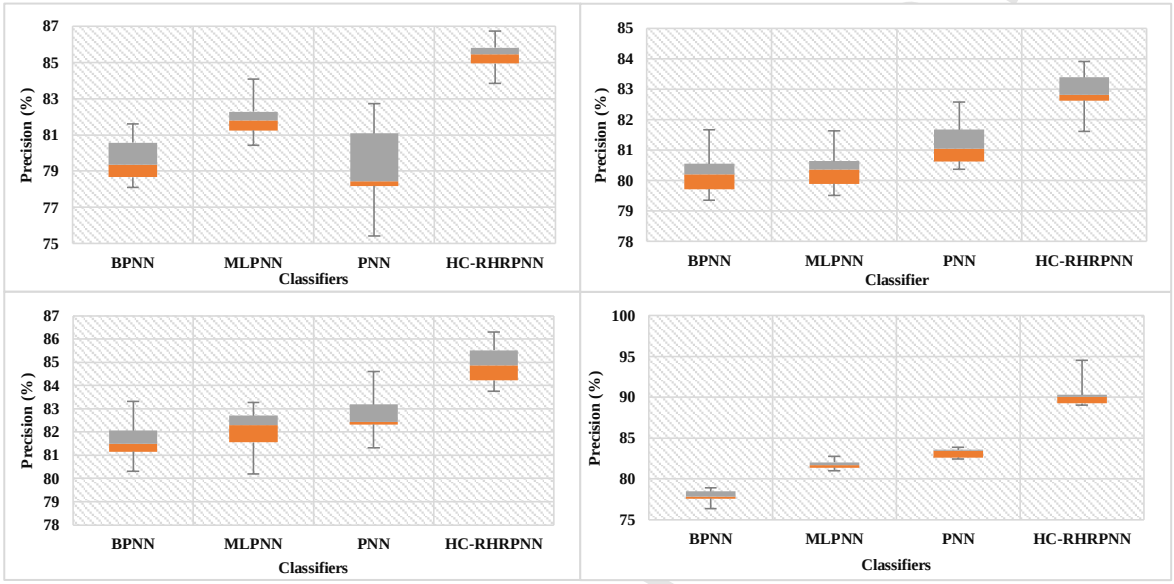


Figure 15: Precision Distribution in Different Ratio of Training Datasets (a) Training dataset ratio=50%; (b) Training dataset ratio=60%; (c) Training dataset ratio=70%; (d) Training dataset ratio=80%

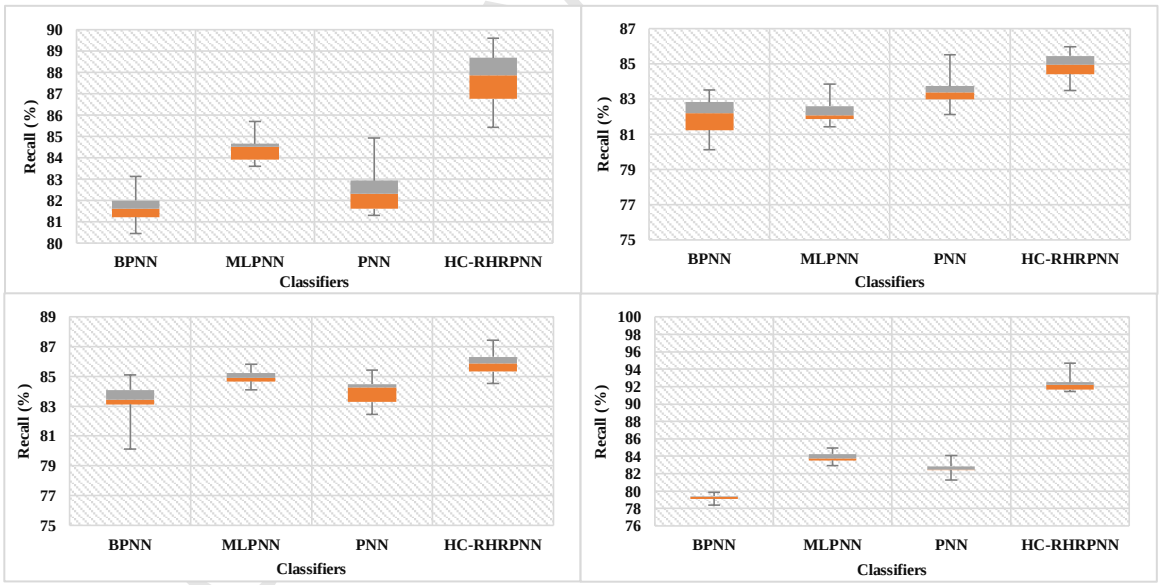


Figure 16: Recall Distribution in Different Ratio of Training Datasets (a) Training dataset ratio=50%; (b) Training dataset ratio=60%; (c) Training dataset ratio=70%; (d) Training dataset ratio=80%

As reported in Figure 14-17 (a-c), HC-RHRPNN outperforms PNN, BPNN, and MLPNN in terms of performance (accuracy, precision, recall, and F-Score) and stability for 50%, 60%, 70% of the training dataset ratio. Further, while considering 80% of D_{Train}, the stability of HC-RHRPNN was found to be similar to the neural network variants, however HC-RHRPNN shows its predominance in terms of performance (Figure 14-17 (d)).

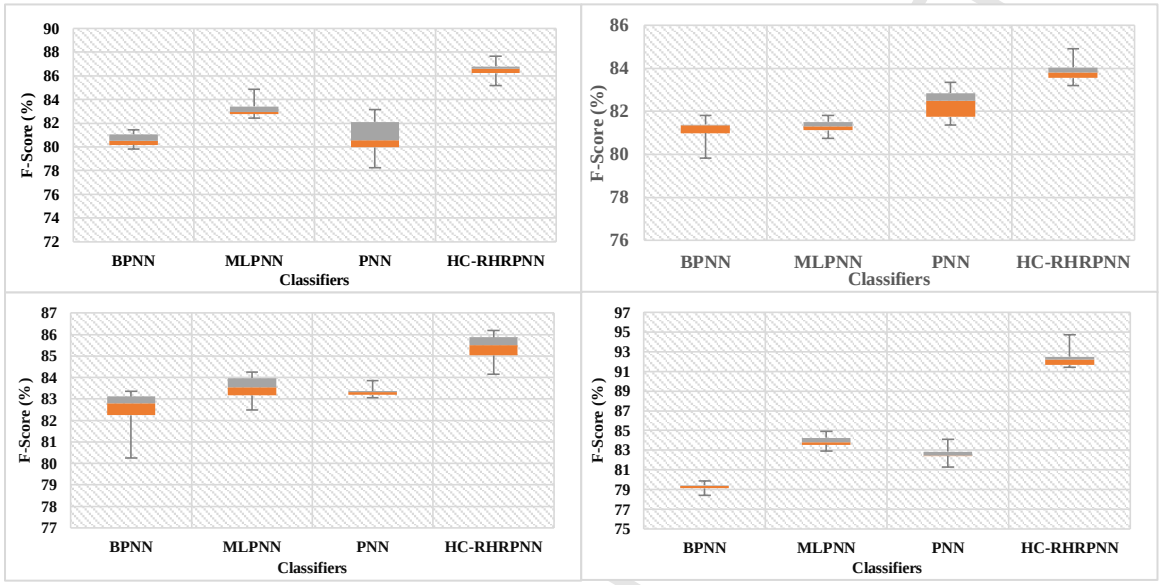


Figure 17: F-Score Distribution in Different Ratio of Training Datasets (a) Training dataset ratio=50%; (b) Training dataset ratio=60%; (c) Training dataset ratio=70%; (d) Training dataset ratio=80%

Further, we employ one-way ANalysis Of Variance (ANOVA) test with a significance value of 0.05 to evaluate the difference between HC-RHRPNN and variants of the neural network in terms of classification accuracy, precision, recall, and F-Score. From Table 8-10, it is clear that the p-value of ANOVA on HC-RHRPNN, PNN, BPNN, and MLPNN for all the ratio of D_{Train} is less than 0.01. Hence, we can state the significance of HC-RHRPNN over the other neural network variants for solving trust prediction problem in cloud service selection environments.

Table 8: One Way ANalysis Of VAriance (ANOVA) test: Accuracy, Precision, Recall, and F-Score - HG-RHRPNN Vs BPNN

Training Dataset Ratio	Accuracy			Precision			Recall			F-Score		
	HG-RPHNN	BPNN	P-Value	HG-RPHNN	BPNN	P-Value	HG-RPHNN	BPNN	P-Value	HG-RPHNN	BPNN	P-Value
50 %	88.76	83.83	1.41901E-10	85.43	79.62	6.12E-10	87.61	81.62	7.25765E-10	86.51	80.61	3.75015E-14
60 %	87.12	84.62	1.15036E-06	82.89	80.21	1.44241E-07	84.87	82.05	2.13891E-06	83.87	81.12	1.33136E-09
70 %	87.62	85.60	0.001198452	84.94	81.63	3.26298E-07	85.86	83.36	9.26033E-05	85.40	82.49	1.9989E-08
80 %	94.63	81.76	1.43974E-18	90.28	77.82	3.03979E-14	92.33	79.24	6.04823E-19	91.29	78.52	2.54998E-21

Table 9: One Way ANalysis Of VAriance (ANOVA) test: Accuracy, Precision, Recall, and F-Score - HG-RHRPNN Vs MLPNN

Training Dataset Ratio	Accuracy			Precision			Recall			F-Score		
	HG-RPHNN	MLPNN	P-Value	HG-RPHNN	MLPNN	P-Value	HG-RPHNN	MLPNN	P-Value	HG-RPHNN	MLPNN	P-Value
50 %	88.76	86.11	7.2E-06	85.43	81.95	2.59E-07	87.61	84.51	7.41372E-06	86.51	83.21	2.50023E-09
60 %	87.12	85.91	0.002621197	82.89	80.36	1.53529E-07	84.87	82.3	2.11717E-07	83.87	81.32	3.77482E-11
70 %	87.62	86.71	0.06630604	84.94	82.01	1.52581E-06	85.86	84.95	0.012739468	85.40	83.45	1.75162E-07
80 %	94.63	85.62	9.55482E-16	90.28	81.61	5.08673E-12	92.33	83.87	4.96835E-15	91.29	82.72	5.79537E-17

Table 10: One Way ANalysis Of VAriance (ANOVA) test: Accuracy, Precision, Recall, and F-Score - HG-RHRPNN Vs PNN

Training Dataset Ratio	Accuracy			Precision			Recall			F-Score		
	HG-RPHNN	PNN	P-Value	HG-RPHNN	PNN	P-Value	HG-RPHNN	PNN	P-Value	HG-RPHNN	PNN	P-Value
50 %	88.76	84.82	1.32E-09	85.43	79.22	6.5E-07	87.61	82.53	5.44843E-08	86.51	80.84	2.20754E-09
60 %	87.12	85.61	0.000179961	82.89	81.24	2.13891E-06	84.87	83.48	0.002397353	83.87	82.34	2.34778E-05
70 %	87.62	85.10	2.18346E-05	84.94	82.72	5.47512E-05	85.86	84.03	0.000255978	85.40	83.37	1.65699E-09
80 %	94.63	86.11	2.53449E-15	90.28	83.17	1.25575E-10	92.33	82.66	2.34665E-15	91.29	82.91	2.51326E-17

Finally, we have compared the performance of HC-RHRPNN with the recent research contributions in trust prediction models with respect to accuracy and precision. From Table 11, it can be noted that HC-RHRPNN ranks third and first position in terms of accuracy and precision. Nevertheless, it cannot be claimed that HC-RHRPNN outpaces the state-of-the-art techniques in all aspects due to the lack of information on various experimental factors such as sampling method, number of samples, etc. To summarize, the devastating performance of HC-RHRPNN was due to the use of hypergraph coarsening technique for the identification of the informative samples.

Table 11: Comparison with Recent Trust Prediction Models

S.No	Authors	Technique Proposed	Feature Selection	Classification Accuracy (%)	Precision (%)
1.	(Mashinchi et al., 2011)	Fuzzy Linear Regression Analysis (FLRA)	x	n/a	79.46***
2.	(Mao et al., 2017)	Particle Swarm Optimization driven	x	n/a	88.28**

		Neural Network (PSO-NN)			
3.	(Mohanty et al., 2010)	Group Method of Data Handling (GMDH)	✓	100*	n/a
		TreeNet		99.72**	n/a
4.	**Proposed Approach**	**HC-RHRPNN**	✓	**94.63*** **	**90.28***

5. Conclusions

Trustworthiness plays a significant role in determining the quality of the candidate services for the design of efficient and resilient service-based systems. Recent research trends in cloud service selection signifies the impact and popularity of trust based cloud service selection models due to fact that trustworthiness of a service is well-reflected by several functional and non-functional QoS attributes. However, the dynamic nature of service-oriented environments in terms of variable QoS values and emergence of new cloud services affects the performance of trust-based cloud service selection models. Trust prediction, a classification problem can be modeled as a suitable solution for the trust based cloud service selection problem through predicting the trustworthiness of the cloud services based on their relevant historical QoS information. Therefore, several researchers focused towards the development of numerous trust and QoS prediction models based on machine learning and statistical techniques. Artificial Neural Networks (ANN) and its variants have proven their significance in solving service selection problems with high prediction accuracy. However, several challenges related to weights, training time, and kernel functions makes the development of an efficient and stable neural network architecture, an open research challenge.

Hence, this work presents a multi-level Hypergraph Coarsening based Robust Heteroscedastic Probabilistic Neural Network (HC-RHRPNN) to predict the trustworthiness of services in cloud environments. HC-RHRPNN uses hypergraph coarsening for the identification of informative samples, which were then used to train HRPNN to achieve high prediction accuracy and to minimize the runtime. The predominance of HC-RHRPNN was proved with a set of extensive experiments on Quality of Web Service (QWS), a public QoS dataset in terms of classifier accuracy, precision, recall, and F-Score. Further, HC-RHRPNN was found to be adaptive, efficient, scalable, and applicable for classification and prediction problems in intrusion detection systems, energy prediction, stock market analysis, medical diagnosis, and metadata quality analysis.

Acknowledgements

This work was supported by The Department of Science and Technology, India; The Council for Scientific and Industrial Research, India; TATA Realty - SASTRA Srinivasa Ramanujan Research Cell, India (Grant No: DST/INSPIRE Fellowship/2013/963, CSIR-SRF Fellowship/143404/2K15/1, MRT/2017/000155, SR/FST/MSI-107/2015 and SR/FST/ETI-349/2013).

References

Akshya Kaveri, B., Gireesha, O., Somu, N., Gauthama Raman, M. R., & Shankar Sriram, V. S. (2017). E-FPROMETHEE: An Entropy Based Fuzzy Multi Criteria Decision Making Service Ranking Approach for Cloud Service Selection. In S. N. S. Venkataramani G., Sankaranarayanan K., Mukherjee S., Arputharaj K. (Ed.), *Smart Secure Systems – IoT and Analytics Perspective. ICIIT 2017. Communications in Computer and Information Science* (pp. 224–238). Springer, Singapore. https://doi.org/10.1007/978-981-10-7635-0_17

Al-Masri, E., & Mahmoud, Q. H. (2007). QoS-based discovery and ranking of Web services. In *Proceedings - International Conference on Computer Communications and Networks, ICCCN* (pp. 529–534). https://doi.org/10.1109/ICCCN.2007.4317873

Ali Sunyaev, S. S. (2013). Cloud Services Certification. *ACM Communications*, *56*(2), 33–36. https://doi.org/10.1145/2408776.2408789

Berge, C., & Minieka, E. (1973). *Graphs and hypergraphs*.

Bostani, H., & Sheikhan, M. (2017). Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept. *Pattern Recognition*, *62*, 56–72. https://doi.org/10.1016/j.patcog.2016.08.027

Çatalyürek, Ü. V., & Aykanat, C. (1999). Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, *10*(7), 673–693. https://doi.org/10.1109/71.780863

Chen, Z., Shen, L., Li, F., & You, D. (2017). Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service QoS prediction. *Knowledge-Based Systems*, *138*, 188–201. https://doi.org/10.1016/j.knosys.2017.10.001

David Nuñez-Gonzalez, J., Graña, M., & Apolloni, B. (2015). Reputation features for trust prediction in social networks. *Neurocomputing*, *166*, 1–7. https://doi.org/10.1016/j.neucom.2014.10.099

Fang, H., Guo, G., & Zhang, J. (2015). Multi-faceted trust and distrust prediction for recommender systems. *Decision Support Systems*, *71*, 37–47. https://doi.org/10.1016/j.dss.2015.01.005

Fang, H., & Yousef Saad. (2011). *Multilevel Linear Dimensionality Reduction using Hypergraphs for Data Analysis*. *Minnesota Supercomputer Institute,University of Minnesota*. Minneapolis, MN.

Fu, Y., Hu, Z., & Zhang, Q. (2008). Bayesian Network based QoS Trustworthiness Evaluation Method in Service Oriented Grid. *2008 The 9th International Conference for Young Computer Scientists*, 293–298. https://doi.org/10.1109/ICYCS.2008.390

Gauthama Raman, M. R., Kirthivasan, K., & Shankar Sriram, V. S. (2017). Development of Rough Set – Hypergraph Technique for Key Feature Identification in Intrusion Detection Systems. *Computers & Electrical Engineering*, *59*, 189–200. https://doi.org/10.1016/j.compeleceng.2017.01.006

Gauthama Raman, M. R., Somu, N., Kirthivasan, K., Liscano, R., & Shankar Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and

feature selection in support vector machine. *Knowledge-Based Systems*, *134*, 1–12. https://doi.org/10.1016/j.knosys.2017.07.005

Han, S., & Cho, S. (2005). Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man, and Cybernetics, PART B: Cybernetics*, *36*(3), 559–570. Retrieved from http://ieeexplore.ieee.org/abstract/document/1634649/

Huang, J., Nie, F., Huang, H., Lei, Y., & Ding, C. (2013). Social trust prediction using rank-k matrix recovery. *Proceedings of the Twenty-Thrid International Joint Conference on Artificial Intelligence*, 2647–2653. https://doi.org/10.1145/2541268.2541270

Kang, U. (2011). Centralities in Large Networks : Algorithms and Observations. *SIAM International Conference on Data*, 119–130. https://doi.org/10.1137/1.9781611972818.11

Kuremoto, T., Kimura, S., Kobayashi, K., & Obayashi, M. (2014). Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, *137*, 47–56. https://doi.org/10.1016/j.neucom.2013.03.047

Luo, X., Lv, Y., Li, R., & Chen, Y. (2015). Web Service QoS Prediction Based on Adaptive Dynamic Programming Using Fuzzy Neural Networks for Cloud Services. *IEEE Access*, *3*, 2260–2269. https://doi.org/10.1109/ACCESS.2015.2498191

Ma, H., Zhu, H., Hu, Z., Li, K., & Tang, W. (2017). Time-aware trustworthiness ranking prediction for cloud services using interval neutrosophic set and ELECTRE. *Knowledge-Based Systems*, *138*, 27–45. https://doi.org/10.1016/j.knosys.2017.09.027

Mao, C., Lin, R., Xu, C., & He, Q. (2017). Towards a Trust Prediction Framework for Cloud Services Based on PSO-Driven Neural Network. *IEEE Access*, *5*, 2187–2199. https://doi.org/10.1109/ACCESS.2017.2654378

Mashinchi, M. H., Li, L., Orgun, M. A., & Wang, Y. (2011). The Prediction of Trust Rating Based on the Quality of Services Using Fuzzy Linear Regression. In *IEEE International Conference on Fuzzy Systems (FUZZ),* (pp. 1953–1959).

Mehdi, M., Bouguila, N., & Bentahar, J. (2013). A QoS-based trust approach for service selection and composition via Bayesian networks. *Proceedings - IEEE 20th International Conference on Web Services, ICWS 2013*, 211–218. https://doi.org/10.1109/ICWS.2013.37

Mohanty, R., Ravi, V., & Patra, M. R. (2010). Web-services classification using intelligent techniques. *Expert Systems with Applications*, *37*(7), 5484–5490. https://doi.org/10.1016/j.eswa.2010.02.063

Nguyen, Hien, T., Weiliang, Z., & Jian, Y. (2010). A Trust and Reputation Model Based on Bayesian Network for Web Services. *IEEE International Conference on Web Services (ICWS)*, 251–258. https://doi.org/10.1109/ICWS.2010.36

Qu, L. (2016). *Credible Service Selection in Cloud Environments*. Macquarie University.

Raj, E. D., & Babu, L. D. D. (2017). Probabilistic Reputation Features. *Neurocomputing*, *219*, 412–421. https://doi.org/10.1016/j.neucom.2016.09.036

Raman, M., Kannan, K., & Pal, S. (2016). Rough Set-hypergraph-based Feature Selection Approach for Intrusion Detection Systems. *Defence Science*, *66*(6), 612.

Raman, M. R. G., Somu, N., Kirthivasan, K., & Sriram, V. S. S. (2017). A Hypergraph and Arithmetic Residue-based Probabilistic Neural Network for classification in Intrusion Detection Systems. *Neural Networks*, *92*, 89–97. https://doi.org/10.1016/j.neunet.2017.01.012

Sivakumar, A., & Kannan, K. (2009). A novel feature selection technique for number classification problem using PNN — A plausible scheme for boiler flue gas analysis. *Sensors and Actuators B : Chemical*, *139*, 280–286. https://doi.org/10.1016/j.snb.2009.02.015

Somu, N., Kirthivasan, K., & Shankar, S. S. (2017). A computational model for ranking cloud service providers using hypergraph based techniques. *Future Generation Computer Systems*, *68*, 14–30. https://doi.org/10.1016/j.future.2016.08.014

Somu, N., Kirthivasan, K., & Sriram, V. S. S. (2017). A rough set-based hypergraph trust measure parameter selection technique for cloud service selection. *The Journal of Supercomputing*, *73*(10), 4535–4559. https://doi.org/10.1007/s11227-017-2032-8

Somu, N., M.R., G. R., Krithivasan, K., & V.S., S. S. (2018). A trust centric optimal service ranking approach for cloud service selection. *Future Generation Computer Systems*, *86*, 234–252. https://doi.org/10.1016/j.future.2018.04.033

Somu, N., Raman, M. R. G., Kirthivasan, K., & Sriram, V. S. S. (2016). Hypergraph Based Feature Selection Technique for Medical Diagnosis. *Journal of Medical Systems*, *40*(11), 239. https://doi.org/10.1007/s10916-016-0600-8

Sosinsky, B. (2010). *Cloud Computing Bible*. John Wiley & Sons.

Specht, D. (1988). Probabilistic neural networks for classification, mapping, or associative memory. *IEEE International Conference on Neural Networks*, *1*(24), 525–532.

Specht, D. (1990). Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. *IEEE Transactions on Neural Networks*, *1*(1), 111–121.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks*, *3*(1), 109–118. https://doi.org/10.1016/0893-6080(90)90049-Q

Su, K., Xiao, B., Liu, B., Zhang, H., & Zhang, Z. (2017). TAP: A personalized trust-aware QoS prediction approach for web service recommendation. *Knowledge-Based Systems*, *115*, 55–65. https://doi.org/10.1016/j.knosys.2016.09.033

Sun, L., Dong, H., Hussain, F. K., Hussain, O. K., & Chang, E. (2014). Cloud service selection: State-of-the-art and future research directions. *Journal of Network and Computer Applications*, *45*, 134–150. https://doi.org/10.1016/j.jnca.2014.07.019

Tang, M., Dai, X., Liu, J., & Chen, J. (2017). Towards a trust evaluation middleware for cloud service selection. *Future Generation Computer Systems*, *74*, 302–312. https://doi.org/10.1016/j.future.2016.01.009

Venkatesh, S., & Gopal, S. (2011). Robust Heteroscedastic Probabilistic Neural Network for multiple source partial discharge pattern recognition–Significance of outliers on classification capability. *Expert Systems with Applications*, *38*(9), 11501–11514.

Witten, Ian H., Eibe Frank, Mark A. Hall, C. J. P. (2016). *Data Mining: Practical Machine Learning Tools and*

Techniques, Second Edition. Morgan Kaufmann.

Wu, H., Yue, K., Li, B., Zhang, B., & Hsu, C. H. (2017). Collaborative QoS prediction with context-sensitive matrix factorization. *Future Generation Computer Systems*, *82*, 669–678. https://doi.org/10.1016/j.future.2017.06.020

Xu, Y., Yin, J., Deng, S., N. Xiong, N., & Huang, J. (2016). Context-aware QoS prediction for web service recommendation and selection. *Expert Systems with Applications*, *53*, 75–86. https://doi.org/10.1016/j.eswa.2016.01.010

Yahyaoui, H. (2012). A trust-based game theoretical model for Web services collaboration. *Knowledge-Based Systems*, *27*, 162–169. https://doi.org/10.1016/j.knosys.2011.10.014

Yang, Z., & Chen, S. (1998). Robust maximum likelihood training of heteroscedastic probabilistic neural networks. *Neural Networks*, *11*(4), 739–747.

Yang, Z., Zwolinski, M., & Chalk, C. (2000). Applying a robust heteroscedastic probabilistic neural network to analog fault detection and classification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *19*(1), 142–151.

Yu, C., & Huang, L. (2016). A Web service QoS prediction approach based on time- and location-aware collaborative filtering. *Service Oriented Computing and Applications*, *10*(2), 135–149. https://doi.org/10.1007/s11761-014-0168-4

Zhang, W., Wu, B., & Liu, Y. (2016). Cluster-level trust prediction based on multi-modal social networks. *Neurocomputing*, *210*, 206–216. https://doi.org/10.1016/j.neucom.2016.01.108

Zolfaghar, K., & Aghaie, A. (2012). A syntactical approach for interpersonal trust prediction in social web applications: Combining contextual and structural data. *Knowledge-Based Systems*, *26*, 93–102. shttps://doi.org/10.1016/j.knosys.2010.10.007