Chapter 5

Heteroscedastic Gaussian Process Trust Model for Reputation Sources with Independent Error Distribution

Introduction

Reputation-based trust systems are deployed as a popular approach to mitigate the risk by giving a prediction of the degree to which the service provider can be trusted.

Reputation is based on the recommendations provided by other service users. The collection, aggregation and distribution of recommendations about a service provider to be trusted, that can, in turn, be used to characterise and predict that provider's future is the major parts of any reputation system. Reputation is context-dependent and relies on contextual information to give the data a meaning. There is no common definition of context used by researchers. While estimating reputation, more recent transaction behaviour should have a greater impact on a peer's score than older transactions, e.g. weights or aging factors can be used to give more importance to recent experience. Therefore, in many trust and reputation systems [21],[110] the time at which the feedbacks were measured is incorporated into modelling of the reputation. Reputation is a statistical value about the trust probability derived from the behaviour history. While estimating this probability, the prediction variance plays a major role to counter the effect of malicious feedbacks. Prediction variance depicts how much the feedback may deviate from the real

reputation or trust value. In [111], the feedback is considered as a tuple $\langle z, c \rangle$; where z is the feedback value, and c is the prediction variance. Values received from different sources are aggregated in the feedback system using a Kalman filter. Kalman filter also produces the prediction variance. This variance is used to predict the reputation of the target node. In [121], their study argued that this system might not give good result when the correlation is less between different observed samples. So a proper mechanism for selecting the feedbacks in the final aggregation is necessary. Again, the age of the feedback and credibility of the feedback source are not considered in the final aggregation. Incorporating these two factors into the aggregation process will help the reputation evaluator to take care of the changing nature of the provider behavior and malicious intent of the dishonest recommenders. Further, sources of reputation feedbacks are independent of each other. So the amount of noise introduced in the feedback cannot be modeled by a single noise distribution. We propose, in this chapter, a kernel based mechanism for collecting and weighting the feedbacks from independent sources by their age and by the trust value of the sources using Gaussian Process Regression (GPR).

Finally, hypothesis testing method of [113] is used to build a multilayered mechanisms to filter malicious feedbacks.

5.1 Heteroscedastic GP

We keep the notation of section 3.2.3 for convenience and recall the definitions. There are observed target values y_i, given by the true underlying function value f and some i.i.d. noise $\varepsilon \sim N(0, \sigma_n^2)$ as $y_i = f(\mathbf{x}_i) + \varepsilon, \mathrm{y}_i \in R$ Given the data sets of n training inputs $D = \{x_i, y_i\}_{i=1}^n$ and test point x_* the Gaussian process predictive distribution given by

$$\bar{f}_* = k_*^T (K + \sigma_n^2 I)^{-1} y = k_*^T \alpha \tag{5.1}$$

$$\text{cov}(f_*) = k(x_*, x_*) - k_*^T \left[K + \sigma_n^2 I \right] k(x_*) \tag{5.2}$$

From these equations, it can be seen that the noise is independent of the inputs with the value σ_n^2 . Such GP model is termed as *homoscedastic* GP. While homoscedastic GP model is still able to correctly estimate the mean, it totally fails to approximate the variance for the system where the input itself is also noisy. If the global noise level is too small, the variance is underestimated at the beginning. On the other hand, if the noise level is too big, the variance is overestimated at the end. Taking input-dependent noise into account the problem can be solved. Such models are said to be *heteroscedastic*. In such a system where the input, x is a noisy measurements of the actual input, $\bar{x}$ we model :

$$x = \bar{x} + \varepsilon_x \quad \varepsilon_x \sim N(0, \Sigma_x)$$

Here each input dimension is independently corrupted by a zero-mean Gaussian noise, so Σ_x is diagonal. Due to Σ_x, now the level of noise varies at different locations of the input x. Under this assumption, the Eq(5.1)-(5.2) are respectively changed to

$$\bar{f}_* = k_*^T (K + \Sigma_x)^{-1} y = k_*^T \alpha \tag{5.3}$$

$$\text{cov}(f_*) = k(x_*, x_*) - k_*^T \left[K + \Sigma_x \right]^{-1} k(x_*) \tag{5.4}$$

However, unlike the standard GP case, heteroscedasticity leads to an intractable integral for the posterior updates. There are approximate solutions for such GP models [114], [115], [116]. A tractable solution under the assumption of independency between the noise present in all dimensions of the input x, is given in [113].

5.2 Proposed Model

5.2.1 Problem Formulation

We start with some definitions to explain the proposed reputation model.

Definition 1: The reputation feedback is the reputation information of the service provider collected by the evaluator from a third party source.

Here the evaluator is a service user. Evaluation of the reputation is based on the interactions carried out directly between the service provider and the evaluator (direct experience) and the recommendations made by other evaluators. This evaluation finally generates reputation as a statistical value. Let us denote reputation evaluation as a 3-tuple $\langle\langle R\rangle, P, t\rangle$, where $\langle R\rangle$ is the predicted reputation value during the evaluation, P is the reputation prediction variance, given by the square error between the predicted reputation value $\langle R\rangle$ and the real reputation R and t is the time instance at which reputation evaluation is done. When a tuple $\langle\langle R\rangle, P, t\rangle$ is obtained by an evaluator, it can be used either to form a decision for interaction with the service provider or to communicate to other peer service users as reputation feedback. Depending on the nature of the evaluator, the feedback might not be exactly same as the tuple $\langle\langle R\rangle, P, t\rangle$. In other words, if the evaluator is a dishonest one, the feedback he communicates will be different from the true value $\langle\langle R\rangle, P, t\rangle$. So we denote the feedback as $\langle z, c, t\rangle$. Now we formally define a feedback session.

Definition 2: A feedback session is a mapping from $\langle\langle R\rangle, P, t\rangle$ to $\langle z, c, t\rangle$.

In this mapping, z is coming from $\langle R\rangle$ and c from P. Here c is the feedback variance, which serves as the measure of reliability of the feedback. At the same time, it is an indication to other evaluators about how to intelligently aggregate the feedback reputation value z. While evaluating the reputation of a service provider, feedbacks are continually collected through feedback sessions. A feedback received at a session s is denoted $f_s = \langle z, c, t\rangle_s$. Here we assume that $t \le s$ because the feedback received in a session might have been formed earlier than k . After each reception of a feedback f_s, the evaluator tries to predict the real-time reputation R_s of the service provider. During the prediction process

we need to consider the changing nature of the service behavior over time and the reliability of the feedback source. Therefore, the age of the feedback and the credibility of the feedback source are to be accounted for. For this purpose we introduced accounted reputation feedback.

Definition 3: An accounted reputation feedback is the reputation value of the service provider generated by the evaluator, from the reputation feedback, after considering the age of the feedback, and the trustworthiness of the source.

The accounted reputation feedback value is used to predict the real reputation of the service provider in our model. For a feedback $f_s = \langle z, c, t \rangle_s$ from a source *i*, the accounted reputation feedback is calculated as

$$r_s = e^{-k\Delta t} * \mathrm{T}_i * \mathrm{z} \tag{5.5}$$

where $\Delta t = s - t$ and k in the range $[0,1]$ represents the exponential decay factor and T_i is the trustworthiness of the feedback source. Decay rate for various values on k is shown in the Figure 5.1.

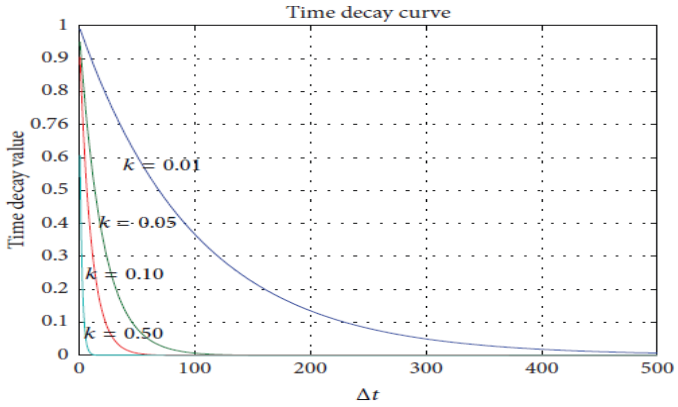

Figure 5.1: Effect of time decay factor k.

It can be observed that aged reputation feedback will have less contribution to the accounted reputation feedback and that less trustworthy source will have less influence on it too.

Definition 4: The reputation history D is a collection of 3-tuples. Each 3-tuple is defined as $\langle \langle z, c, t \rangle, T_i, \langle \mathrm{R}_s \rangle \rangle$ where $\langle z, c, t \rangle$ is the reputation feedback value, T_i is the trustworthiness of reputation feedback source and $\langle \mathrm{R}_s \rangle$ is the

associated estimated real reputation from reputation feedback $\langle z,c,t \rangle$ in a session s .

It may be mentioned here that in the reputation history the estimated reputation $\langle \mathrm{R}_s \rangle$ may be replaced by the real reputation R_s. The reason is that in an assessment of the reputation of a service provider based on a reputation feedback $\langle z,c,t \rangle$, the obtained estimated reputation $\langle \mathrm{R}_s \rangle$ will help service user to decide whether to interact with the service provider at time s . The service user will interact if $\langle \mathrm{R}_s \rangle$ is within her expected reputation range. For example, if her expectation is 0.5 and $\langle \mathrm{R}_s \rangle \geq 0.5$ then she will interact with the service provider. After the interaction the actual reputation value R_s will be obtained. This value R_s may replace $\langle \mathrm{R}_s \rangle$ in the tuple $\langle \langle z,c,t \rangle, T_i, \langle \mathrm{R}_s \rangle \rangle$ to form the tuple $\langle \langle z,c,t \rangle, T_i, R_s \rangle$.

.**Definition 5:** The reputation query is the reputation feedback $f_s = \langle z,c,t \rangle_s$ which initiates the prediction of the real reputation to obtain $\langle \mathrm{R}_s \rangle$ by using the proposed model.

With these definitions, we now formulate the problem that we want to solve.

Problem Definition: Given the reputation history D and a reputation query f_s, how to predict the real reputation R_s value of the service provider.

5.2.2. Generative Model

Essentially reputation is a statistical value derived from feedbacks from recommenders. Let us assume that a service user wants to evaluate the real-time reputation value R_t of a service at time t. The value of R_t is not known to the evaluator. It can only be predicted by using a prediction model. Let this

predicted reputation value be represented by a tuple $\langle \langle R_t \rangle, P_t \rangle$, where $\langle R_t \rangle$ is the predicted reputation value and P_t is the prediction variance, which is an estimation about the square error between $\langle R_t \rangle$ and the real reputation R_t. The variance P_t can be taken as the evaluator's confidence in her prediction because it is an evaluation about the accuracy of the predicted reputation value $\langle R_t \rangle$. A smaller (larger) value of P_t represents a higher (lower) confidence.

In a reputation session, when another evaluator asks for recommendation at another time s, the first evaluator will map tuple $\langle \langle R_t \rangle, P_t \rangle$ to $\langle z, c, t \rangle_s$ and send as reputation feedback. The recipient evaluator, then tries to predict the real-time reputation R_s from the reputation feedback $f_s = \langle z, c, t \rangle_s$ received from a source and also evaluates the prediction variance P_s. Due to the incomplete knowledge of the recommender (may be honest or dishonest) and dynamic nature of the service behaviour, the feedback reputation value usually has a deviation from the real reputation.

The proposed system models the deviation as a zero mean Gaussian noise. In other words, the reputation feedback z is defined as a noisy measurement of the real reputation contaminated by a zero-mean normal Gaussian distribution i.e.

$$z_s = R_s + q_s; \qquad q_s \sim N(0, c_s^{-1}) \tag{5.6}$$

By this formulation, reputation feedback of an untrustworthy estimate is downgraded by increasing its uncertainty proportionally to c_s.

5.2.3 Model prediction

Let D be the reputation history of size n available at time s when a new reputation query value $f_s = \langle z, c, t \rangle_s$ is obtained from a reputation source. Our

interest is to predict (recover) the value of the function in Eq.(5.7) from this reputation history.

$$\langle R_s \rangle = f(z_S) + \varepsilon; \varepsilon \in N(0, P_s) \tag{5.7}$$

Eq. (5.6) and (5.7) completely represent the system as a *heteroscedastic GP*. The input to the function f is a noisy measurement of the real-time reputation R_s and noise term in z_s represents uncorrelated precisions between individual reputation feedbacks. Equivalently, we are trying to recover the real reputation R_s from the available history of past behavior. This is the general regression setting as we are going to regress $\langle R_s \rangle$ on z_s. To prepare the *heteroscedastic* Gaussian process regression (HGPR), we need to define its mean and covariance *K*.

Since we do not have any information about the provider, we set the mean to 0. The *covariance kernel* is selected as

$$k(z_s, z_{s'}) = \sigma_f \exp(\frac{d(r_s, r_{s'})^2}{2l^2}) \tag{5.8}$$

where σ_f is the signal variance, l is the length scale, $d(.,.)$ is the distance function, r_s and $r_{s'}$ are the accounted reputation feedback values for z_s and $z_{s'}$.

It may be noted that covariance of any two reputation feedback values is calculated in terms of their accounted reputation feedback values generated by using Eq.(5.5).. The covariance function captures the correlation between any two reputation feedback samples in their value domain, time domain and as well as in trust domain. It can be shown that the samples that are collected, very close to each other in any of these domains, are more correlated as their distance $d(r_s, r_{s'})^2$ will be less.

In order to derive the prediction equation for HGPR model, we need to assume the mutual independence between the noise terms in Eq. (5.6) i.e. $q_s \perp q_{s'}$ in

order to have a tractable likelihood [114]. This assumption implies $c_s \perp c_{s'}$. Let $\mathbf{y}$ be the vector of time discounted values of all $\langle R \rangle$ components and $\mathbf{z}$ be the vector of accounted reputation feedbacks computed from all z components in all n tuples in the reputation history D. Under our HGP model, the likelihood of $\mathbf{y}$ is a normal p.d.f. expressed as Eq. (5.9).

$$p(\mathbf{y} \mid f) = N(\mathbf{y} \mid f, q_s) \tag{5.9}$$

Let z_* be the z component in the reputation query $f_s = \{z, c, t\}_s$ about which the real reputation is going to be predicted in the session s and y_* be this corresponding predicted output. Then the joint distribution of y_* and $\mathbf{y}$ is a normal p.d.f. given by Eq. (5.10).

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \left(0, \begin{bmatrix} K(\mathbf{z},\mathbf{z}) + \Sigma_{\mathbf{z}} & K(\mathbf{z}, z_*) \\ K(z_*, \mathbf{z}) & K(z_*, z_*) \end{bmatrix} \right) \tag{5.10}$$

$$\Sigma_{\mathbf{z}} = diag(c_s^{-1}) \tag{5.11}$$

$\Sigma_{\mathbf{z}}$ is the diagonal matrix of the noise terms which defined the variability of each reputation feedback value in D. If we set such noise terms constantly to σ then Eq. 5.10 reduces to standard homoscedastic GP.

The prediction of f can be obtained by conditioning z_* to the set of reported observations $\mathbf{z}$ and $\mathbf{y}$ in the reputation history D. By the marginalization properties of the Gaussian distributions, the predictive distribution of $f(z_*)$ at the reputation query point in defined as Eq. (5.12).

$$p(y_* \mid \mathbf{z}, \mathbf{y}, z_*) = N(E[y_*], \sigma^2(y_*)) \tag{5.12}$$

where

$$f_* = E[y_*] = K(z_*, \mathbf{z})[K(\mathbf{z},\mathbf{z}) + \Sigma_{\mathbf{z}}]^{-1}\mathbf{y} \tag{5.13}$$

$$\text{cov}(f_*) = \sigma^2(y_*) = K(z_*, z_*) - K(z_*, \mathbf{z})[K(\mathbf{z}, \mathbf{z}) + \Sigma_{\mathbf{z}}] \tag{5.14}$$

It must be noted that $K(z_*, z_*)$, $K(z_*, \mathbf{z})$, $K(\mathbf{z}, \mathbf{z})$ measure the correlations on accounted reputation feedback values. These are defined as

$$K(z_*, z_*) = k(r_*, r_*) \tag{5.15}$$

$$K(z_*, \mathbf{z}) = [k(r_*, r_1), ..., k(r_*, r_n)] \tag{5.16}$$

$$K(\mathbf{z}, \mathbf{z}) = \begin{bmatrix} k(r_1, r_1) & ... & k(r_1, r_n) \\ ... & ... & ... \\ k(r_n, r_1) & ... & k(r_n, r_n) \end{bmatrix} \tag{5.17}$$

where r_i is the discounted reputation feedback value calculated using Eq. (5.5) from the corresponding reputation feedback value z_i of the i^{th} tuple in D.

.

5.2.4 Parameter Training

A Gaussian process can represent $f(z)$ obliquely, but rigorously, by letting the data 'speak' more clearly for themselves. As such, GP is not completely free from parameters. Eq. (5.13)-(5.14) are conditioned on the set of hyperparameters $\Theta = (\sigma_f, l)$. Since their values are typically unknown, they need to be estimated from the reputation history as a part of model selection. They can be estimated by marginal likelihood optimization method, which sets their values by maximizing the evidence of observations in the reputation history according to the marginal likelihood of the model.

Using the marginalization property of the Gaussian distribution, the log marginal likelihood of the predictive density of GP is defined by Eq. (5.18).

$$L(p(f \mid \mathbf{z}, \mathbf{y}, z_*) = \ln(f(\mathbf{y} \mid f, \mathbf{z}) p(f \mid \mathbf{z}) df = -\frac{1}{2}\mathbf{y}^T C^{-1}\mathbf{y} - \frac{1}{2}\ln(C) - \frac{n}{2}\ln(2\pi) \tag{5.18}$$

where $C = K(\mathbf{z},\mathbf{z}) + \Sigma_{\mathbf{z}}$. Taking the partial derivatives of Eq. (5.18) over $\Theta = (\sigma_f, l)$ we can obtain the following.

$$\frac{\partial L}{\partial \Theta} = \frac{1}{2}\mathbf{y}^T C^{-1}\frac{\partial C}{\partial \Theta}C^{-1}\mathbf{y} + \frac{1}{2}tr(C^{-1}\frac{\partial C}{\partial \Theta}) \tag{5.19}$$

Now factoring in the expression of the kernel of Eq.(5.5) we can obtain the following equations.

$$\frac{\partial L}{\partial \sigma_f} = 2\sigma_f \exp(-\frac{d^2}{2l^2})) \tag{5.20}$$

$$\frac{\partial L}{\partial l} = -\frac{\sigma_f^2 d^2}{l^3}\exp(-\frac{d^2}{2l^2}) \tag{5.21}$$

The set of the values of the hyperparameters is:

$$\Theta_{ML} = \left\{\sigma_{f,ML}, l_{ML}\right\} = \arg\max_{\sigma_f, l}(\ln(p(\mathbf{y} \mid \mathbf{z}, \boldsymbol{\theta}, \sigma_f, l) \tag{5.22}$$

where $\boldsymbol{\theta} = \{c_1, ..., c_n\}$ is the vector of all c_i's from D.

The standard Polak-Ribiere conjugate descend method provided by the GPML Matlab toolbox can be used to solve Eq. (5.22) and hence the hyper parameters can be estimated. The training algorithm is given in Algorithm 5.1.

Algorithm 5.1: HGP training

Inputs:

y : time discounted predicted reputation vector.
z : accounted reputation feedback vector
σ_f^0 : Initial guess of signal noise
l^0 : Initial guess of the length scale.
err : Error bound.
max *Iter* : Maximum no. of iterations.

Outputs:

$\Theta^{\max Iter}$: Estimated hyperparameters

Steps:

1. $\Theta^0 \leftarrow \left\langle \sigma_f^0, l^0 \right\rangle$;
2. $\gamma^0 \leftarrow -\frac{\partial}{\partial \Theta}(\ln(p(\mathbf{y} \mid \mathbf{z}, \Theta^0))$;
3. $h \leftarrow 0$;
4. **while (**$\left|\Theta^{h-1} - \Theta^h\right| \prec err \,\&\&\, h \prec \max Iter$**) do**
5. **Begin**
6. $h \leftarrow h + 1$;
7. $\Delta\Theta^h \leftarrow -\frac{\partial}{\partial \Theta}(\ln(p(\mathbf{y} \mid \mathbf{z}, \Theta^{h-1}))$;
8. $\beta^h \leftarrow \frac{\left(\Delta\Theta^h\right)^T \left(\Delta\Theta^h - \Delta\Theta^{h-1}\right)}{\left(\Delta\Theta^{h-1}\right)^T \Delta\Theta^{h-1}}$
9. $\gamma^h \leftarrow \Delta\Theta^{h-1} + \beta^h \gamma^{h-1}$;
10. $\alpha^h \leftarrow \arg\max_\alpha p\left(\mathbf{y} \mid \mathbf{z}, \left(\Theta^{h-1} + \alpha\gamma^{h-1}\right)\right)$
11. $\Theta^h \leftarrow \Theta^{h-1} + \alpha^h \gamma^h$;
12. **End**
13. **return** $\Theta^h \leftarrow \left\langle \sigma_f^h, l^h \right\rangle$;

After initializing the hyperparameters in steps 1, the conjugate gradient loop from steps 4-12 computes the gradient with respect to the hyperparamets of the previous iteration and line search direction given by β and α parameters. When the algorithm terminates, it returns the values of the hyperparameters in the last iteration.

5.2.5 Malicious feedback Detection Mechanism

Once the HGPR model is trained prediction can be done from Eq.(5.13)-(5.14). To make the model more robust against the malicious feedbacks, we built into it the last line of defence based on the hypothesis testing method of [111]. We briefly describe the method in the following.

Let H_0 be the hypothesis that reputation feedback z of a reputation query $f_s = \langle z, c, t \rangle_s$ is honest. Eq. 5.14 provides the prediction variance $\sigma^2(y_*)$. In an environment without malicious recommenders, the deviation between the prediction f_* and z must follow a zero-mean normal distribution with variance $\sigma^2(y_*) + Q_s$ where Q_s is yielded by Eq. (5.23) from the all predictions up to session s.

$$Q_s = \frac{1}{n}\sum_{i=1}^{n} z_i - \langle R_s \rangle^2 \tag{5.23}$$

Hypothesis testing method is to test whether the deviation between the f_* and z is normal enough. For a desired confidence level of the test δ, the hypothesis testing is to find the a threshold value $maliTH$ such that

$$P(|\, z - f_* \,| \geq maliTH) = \delta \tag{5.24}$$

Under the hypothesis H_0, the deviation, $(z - f_*)$ follows a zero-mean normal distribution with the variance $\sigma^2(y_*) + Q_s$, so we can also have that

$$P(|\, z - f_* \,| \geq maliTH \mid H_0) = 2 \times \Pi(maliTH / \sqrt{\sigma^2(y_*) + Q_s} \tag{5.25}$$

where $\Pi(.) = 1 - \phi(.)$, with $\phi(.)$ being the cumulative distribution function of a zero-mean unit variance normal distribution. Solving Eq. (5.24) –(5.25), we get that

$$maliTH = \sqrt{\sigma^2(y_*) + Q_s}\,\Pi^{-1}(\delta / 2) \tag{5.26}$$

If the deviation $(z - f_*)$ exceeds the threshold $maliTH$ then the hypothesis H_0 is rejected and therefore the feedback z of f_s is reported as malicious.

5.2.6. Final Algorithm

After having described the training algorithm and malicious detection mechanism, we now present the final algorithm for our proposed heteroscedastic Gaussian Process trust model for reputation sources with independent error distribution (HGPTrust). We know present our prediction model in the form of an algorithm:

Algorithm 5.2: HGPTrust.

Inputs:

D: Reputation history.

Output:

$isMali$: Malicious indication.

f_*: Predicted reputation value.

D: Reputation history after update.

Steps:

1. $\mathbf{y} \leftarrow GenYvec(D)$;
2. $\mathbf{z} \leftarrow GenZvec(D)$;
3. $f_s \leftarrow \langle z, c, t \rangle_s$; [Reputation query]
4. $T \leftarrow getTrust()$; [Trust of the source of reputation query]
5. $r_* \leftarrow z \times T \times e^{-k(s-t)}$; [From Eq. (5.5)]
6. Train the HGP using Algorithm 5.1;
7. Obtain the predicted value f_* using Eq.(5.13);
8. Obtain the prediction variance $\sigma^2(y_*)$ using Eq.(5.14);
9. Calculate Q_s using Eq.(5.23);
10. Calculate $maliTH$ using Eq.(5.26);
11. **if** $(z - f_*) \succ$ maliTH **then**
12. $isMali \leftarrow true$;
13. **else**
14. $isMali \leftarrow false$;
15. Insert $\langle \langle z, c, t \rangle_s, T, f_* \rangle$ to D;
16. **endif**
17. return $isMali$, f_*, D;

First we generate the accounted reputation feedback vector $\mathbf{z}$ and time discounted predicted reputation vector $\mathbf{y}$ using tuples of reputation history D. Next reputation feedback from a reputation source is collected at time s in step 3. The trustworthiness value of the reputation source is then obtained from the environment. Training and subsequent prediction are done in steps 5-9. Using the prediction variance the malicious threshold is then computed in step 10. Using the threshold value, malicious feedback is detected in step 11-12. If the feedback is not malicious, then a new tuple is formed and inserted into the reputation history in step 15. At the end, the malicious flag, predicted value and updated reputation history are returned.

The algorithm requires $O(n^3)$ time to compute the output due to the inversion of the covariance matrix in Step-1 of algorithm 3.1 in Chapter-3. However, after the inversion of the covariance matrix, prediction only takes $O(n)$ time for f_* and $O(n^2)$ for the predictive variance $\sigma^2(y_*)$. In practice, while training our model with 1000 data points, it approximately takes 1.8 minutes on a 4 Core i5 2.3 GHz CPU, 3GB RAM architecture. The time increases as we add more data points to the training set.

5.3 Experiment and Results

5.3.1 Data Set

The model is tested on the Epinions data obtained from the first author of [119]. Epinions is an online community website where users write reviews of products from different categories. Users also read and rate other reviews on a numerical scale based on their usability. Statistics of the data set is reflected in the Table 5.1.

Table 5.1: Statistics of the Dataset [119]

# of Users	22166
# of Products	296277
# of Categories	27
# of Rating	922267
Ave Rating	4.05

In our experiment, we consider a product category analogous to a Web service provider and different products under this category as the Web services provided by this provider (Category). The rating of a product (Web service) from a user is considered as the real reputation value R of the Web service. As reputation feedback value is a nosy measurement of the real reputation value, we, therefore, simulate the reputation feedback by adding a deviation which follows a zero-mean Gaussian distribution. The variance of the distribution is randomly picked up from the set {0.01, 0.02, 0.03}. Further in our model the reputation feedback values from independent sources are associated with independent noise distributions. Therefore, the percentage of rating under these three noise distributions are divided as 20%, 30% and 50%. We use two random integers to pick up the variance value and distribution % value to form random combinations of them. For example, if the value of the first random integer is 2 and that of the second random inter is 1, then we randomly add to 20% of the ratings a deviation which is generated by a zero-mean Gaussian distribution with variance value equal to 0.02. Finally, the usability value of each rating from a user is considered as the trustworthiness value T of the reputation feedback source. The category we select from the dataset is 1 which is “Online Stores and Service”. The product number 122 (no information for its name is available) under this category is selected in the experiments. There are 921343 ratings available for this product in the dataset.

5.3.2 Experimental Results

We perform two experiments to evaluate the prediction accuracy and robustness of the malicious detection mechanism. We compare our model against RLM [111]. In the first experiment, we do not use the malicious detection mechanism in both the models and compare the prediction accuracy by using the following performance indicators.

$$Mean\,Square\,Error(MSE) = \frac{1}{n}\sum_{t=1}^{n}(R_i - \langle R_i \rangle)^2 \tag{5.27}$$

$$Mean\,Absolute\,Error(MAE) = \frac{1}{n}\sum_{t=1}^{n}\left|R_i - \langle R_i \rangle\right| \tag{5.28}$$

$$Symetric Mean\,Absolute\,Percentage\,Error(SMAPE) == \frac{100}{n}\sum_{t=1}^{n}\frac{\left|R_i - \langle R_i \rangle\right|}{R_i + \langle R_i \rangle} \tag{5.29}$$

where R_i is the real reputation and $\langle R_i \rangle$ is the predicted real reputation.

For booting our model we require the reputation history D. As reputation history D, we take from the dataset, the first 90 combinations of noise contaminated rating, variance value used in generating the added noise, time point at which the rating has been created, usability value of the rating and corresponding value of original rating. These values, in order, play the roles of z, c, t, T and $\langle R \rangle$ of a tuple in the history D.

In the dataset, the usability rating is done by using an integer in the range 1 to 5. Since, we use this usability value as the trust of the feedback source,we convert it into a real values as $\{1 \to 0.1, 2 \to 0.25, 3 \to .5, 4 \to 0.75, 5 \to 1\}$. Reputation queries are continuously constructed from the 99th record onwards and 1000 predictions are made using them. The RLM model is also executed from the 101th data point onwards. Every time a prediction is made our reputation history is extended by one more tuple as explained in step 15 of

Algorithm 5.2. To incorporate the time dependent behavior of the Web service reputation, we use the exponential time decay principle used in Chapter 4. The most recently added 100 points in D are decayed with a k value of 0.01 and older points are decayed with k value of 0.02 in our experiments. Thus we give more importance to the latest reputation sessions that older sessions. The results of the prediction performance by the models are given in Table 5.2.

Table 5.2: Error measurements for 1000 prediction sessions: MSE, MAE and SMAPE

	MSE			MAE			SMAPE		
	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.
HGPTrust	0.0	2.3	1.02	0.0	1.2	0.75	0.0	8.0	5.34
RLM[111]	0.82	4.54	3.86	0.0	12.03	8.89	0.0	13.03	9.23

We can observe from Table 5.2 that our model has better values of the metrics compare to the RLM model. The predictions from RLM model are noisy. One reason of the noisy prediction of RLM is due to the effect of having a single noise variance parameter in the model.

In our next experiment, we compare the robustness of the two models in the presence of malicious feedbacks. We simulate a malicious feedback in this way. The raters whose usability value is less than or equal to 0.25 are considered as low trustworthy recommenders. We selected 10%, 20% and 30% of these users at random and the true variance (any value from the set {0.01, 0.02, 0.03}) values used for the generation of the noise components in their ratings are replaced by a very low value i.e.10^{-4}. The principle behind this setting is that these raters provide a very low value of variance in their reputation feedbacks to increase the precision level of their feedbacks.

The feedbacks from these selected untrustworthy recommenders are the positive feedbacks that the malicious detection mechanism must filter. Whereas the rest are the negative feedbacks that must be accepted. Now we define the following.

False positive: A negative feedback that has been reported as malicious

True positive: A positive feedback that has been correctly marked as malicious.

We compare the two models by using false positive rate (FPR) and true positive rate (TPR). They are defined as:

$$FPR = \frac{No.\ of\ \ false\ positives(nfp)}{No.\ of\ \ negative\ reputation\ feedbacks\ in\ the\ dataset(nf)} \tag{5.30}$$

$$TPR = \frac{No.\ of\ true\ positives(ntp)}{No.\ of\ \ positive\ reputation\ feedbacks\ in\ the\ dataset(np)} \tag{5.31}$$

We use $\delta = 5$ in the experiment. The results obtained in the three runs of the experiment for different proportion of the untrustworthy recommenders are presented in the Figure.5.2.

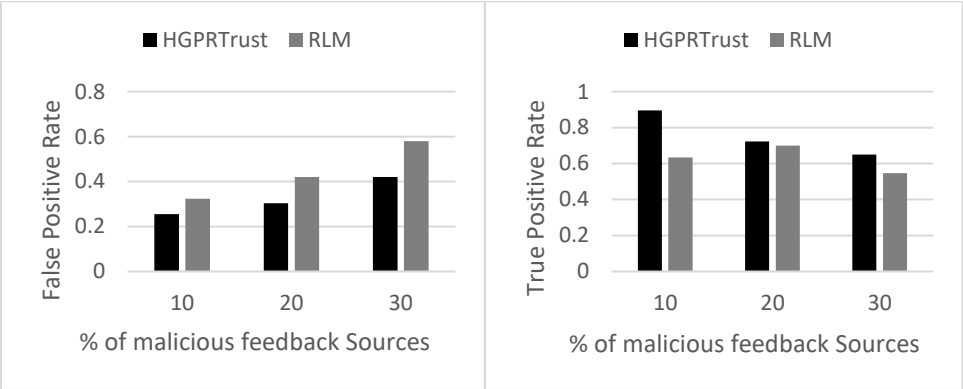

Figure 5.2: False positive Rate and True Positive Rate

Figure.5.2 shows that the detection mechanism of our model is much better than that of RLM. Specifically when the number of malicious feedback sources is 10%, HGPRTrust has a significant FPR value of 0.255 and higher TPR value of 0.895. However when the number of malicious raters increases, the advantage of HGPRTrust decreases as we can see in the bar graphs of 20% and 30% of malicious feedback sources.

The duration of a prediction of the proposed model is shown in the Figure 5.2 for the 1000 training points. Computational time is plotted with respect to the number of training examples. The plot shows that the computation time

requirement of GP based regression model rises fast with the size of the training set as expected.

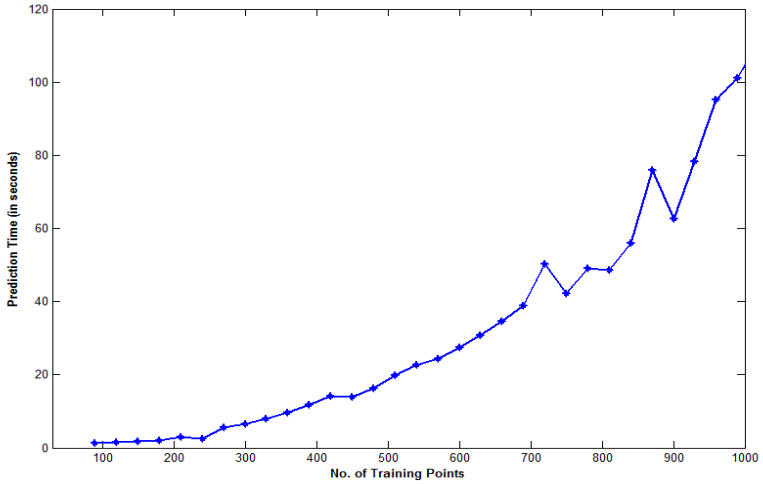

Figure 5.3: Time in second needed for prediction of one query point.

So the application of the proposed model on the online scenario is computationally expensive especially when the size of reputation history D becomes larger. One remedy is to use a moving window, so that the number of points in the history D is always bounded by the size of the window. Another one can be to localize the GP by learning local linear structure present in the sequential feedback sessions [156]. Once such structures are identified, each local region represented by a subset of the history D with lesser number of training points can be used to fit a version of the proposed model so that computational time can be reduced. The final prediction can be obtained by taking the average of all local models selected by the reputation query.

5.4 Conclusion

We propose a time and trust aware reputation prediction framework using the heteroscedastic Gaussian process model. The model is found to be effective to handle the time variant and independently noise contaminated reputation values. In other words, the model is able to tackle the problem of dealing with

heterogeneous data reliabilities in reputation evaluation by using trust parameters i.e. trustworthiness and level of confidence in the feedback for the recommenders to scale the data noise rates of the HGP. In this way, the model has the ability to flexibly increase the noise around reports associated with untrustworthy users. Then, by training the model with the feedbacks gathered from the recommenders, we are able to evaluate a feedback and learn its trustworthiness. We show that our method is more accurate than other exiting model with an extensive experimental evaluation on real-word data.