Predicting Imprecise Failure Rates from Similar Components: a Case Study using Neural Networks and Gaussian Processes

Philipp Limbourg and Hans-Dieter Kochs

Information Logistics, Department of Engineering,
University of Duisburg-Essen
Bismarckstr. 90, 47057 Duisburg, Germany
{philipp.limbourg,hans-dieter.kochs}@uni-due.de
http://www.uni-due.de/il

Abstract: Reliability prediction methods for early design stages suffer from the lack of empirical failure data. However, expert knowledge and data from similar in-service components are information sources ready available. In this work, an expert elicitation procedure for failure rate prediction is presented. In contrast to direct elicitation procedures such as FMEA, relations to in-service components are elicited. Neural networks and Gaussian processes are used to capture relations such as "similarity" and "wear out" to predict failure rate bounds and prediction uncertainty of new components. Both approaches are evaluated on a test set with raw failure rates. Results indicate that the method is a promising alternative to direct elicitation procedures.

1 Introduction and related work

Since the release of the German industry guideline VDI 2206 "Design methodology for mechatronic systems" [VDI04], there has been a noticeable growth in interest for reliability prediction methods over all project phases. Along with the proposition of the V-Modell as a process model covering the whole development cycle, it has been questioned to do reliability evaluation mainly in the integration phase [KP04]. The change in reliability modeling approaches in the very recent past is mainly influenced by the changed needs of mechatronic industries. Project durations have dramatically decreased, and most of the important design decisions are done early in system and domain design phases. Reliability flaws discovered later on could only influence the design along with high costs. Due to the budget which tends to be smaller than it might be in aviation, nuclear industry or military, extensive modeling and testing might not be daily practice. The resulting requirements for reliability analysts can be summarized in the following goal: provide a reliability estimate as soon as possible and with low costs.

Companies most often do not create only one product but product families with a number of variants. If data already acquired was efficiently re-used for new products, this might lead to a drastically improved accuracy/cost ration. That gives the motivation of our work.
Knowledge from similar components is a tremendously important source of evidence in reliability prediction for components in their early development phase. Reliability figures of components already in use may be considered as very precise. Instead of assigning direct estimates on reliability parameters, similarity measures between new and existing components have to be elicited. The more astonishing is that this field is almost totally unnoticed in the reliability community. Much work has been done about direct expert estimates as e.g. FMEA, but the transfer of similar component data to the project is still rather unexplored. This does not mean the knowledge is neglected altogether. In practice, old reliability figures are used, but mainly without any systematic approach.

Few works exist, which cover the inclusion of similar component knowledge into reliability prediction. In [KRB02, HB05] an interesting approach is described and evaluated. The prior probability of a Bayesian testing model is influenced by a "transfer factor". This transfer factor denotes the grade of similarity between old and new designs. The prior degrades to a completely non-informative one if the transfer factor decreases to zero. The method proposed in [Gul99, AJ00] applies similarity analysis to failure rate predictions of avionic components. Deterministic algorithms are used to quantify the new failure rates based on the number of similar building parts between systems. The approach is restricted to constant failure rates which is generally not the case for mechanic components. These approaches share some commonalities. They only treat the grade of "distance", no further estimates may be given by experts. They are restricted to fixed distributions (e.g. Weibull or Exponential) and do not account for prediction uncertainty (except the transfer factor).

In the next section, the elicitation process is covered in detail. Similarity relations are estimated, an imprecise prediction function is formed and training data for regression is constructed. Section 3 shows, how neural networks and Gaussian processes may be used to predict failure rates by solving the imposed regression problem. Section 4 finally evaluates both approaches on their applicability and proposes some further research directions.
2 Scenario

We consider a new component $C_X$ without any known reliability data. Yet, we have similar in-service components $C_A, C_B, C_C$ with known failure rates $\lambda_A(t), \lambda_B(t), \lambda_C(t)$. These failure rates may be empirically derived and thus not representable by analytical distributions. The target is to make a useful prediction of $\lambda_X(t)$ by developing a combined expert elicitation – machine learning method (Fig. 1). This procedure has to make use of the known in-service failure rates and expert estimates about the relations between components. It is divided into two steps: Elicitation of similarity relations and prediction of failure rates using trained neural networks / Gaussian processes.

2.1 Elicitation

The elicitation procedure allows engineers to select components and express their believe about their similarity to $C_X$. First, an engineer has to select components from a database that he believes share some similar properties with $C_X$. These might be reliability characteristics, but also form, shape or functionality. The elicitation procedure does not specify what defines a ”similar” component. Experts are able to do this much more accurate (e.g. similarities in the CAD models or similar stress simulations). A set of qualifiers is provided and engineers state how strong these qualifiers describe the relation between both components. An engineer may estimate three different characteristics: distance to the known component ($q_{dist}$: similar ↔ not similar), reliability in general ($q_{qual}$: better ↔ worse), wear out ($q_{wo}$: higher ↔ lower), e.g.:

"$C_X$ is very similar to $C_A$"

"$C_X$ is moderately better than $C_A$"

"$C_X$ shows higher wear out than $C_A$"

The engineers have to know neither $\lambda_X(t)$ nor its meaning. This is especially beneficial as most engineers are experts in their domain but they may be novices in reliability engineering. In this first study, we are mapping these verbal estimates to real numbers between $-1$ and 1. Further studies may e.g. include a fuzzy or imprecise number approach. Hence, a relation between two components $C_X$ and $C_A$ may be described by a vector $q = (q_{dist}, q_{qual}, q_{wo})^T$. From the knowledge of $q$ and the failure rate of $C_A$ we would now like to generate the failure rate of $C_X$, and thus a regression problem emerges:

$$f_{sim} : \mathbb{R}^5 \rightarrow \mathbb{R}$$

$$f_{sim}(t, \lambda_A(t), q) = \lambda_X(t)$$

$f_{sim}$ represents the experts’ way of expressing statements as ”similar” or ”better” which should be regressed. This expression is by nature not exact, there are many different components $C_X$ fulfilling the property of being ”similar” to $C_A$ according to the expert’s subjective opinion. Thus, $\lambda_X(t)$ becomes an imprecise quantity suffering from prediction uncertainty. Hence, there is no interest in precise predictions but for methods which are
Table 1: Weibull scale ($\alpha$) and shape ($\beta$) parameters.

<table>
<thead>
<tr>
<th>Failure rate</th>
<th>$\alpha$</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>70000</td>
<td>1.8</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>60000</td>
<td>1.6</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>50000</td>
<td>1.4</td>
</tr>
<tr>
<td>$\lambda_4$</td>
<td>40000</td>
<td>1.2</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>30000</td>
<td>1.4</td>
</tr>
<tr>
<td>$\lambda_6$</td>
<td>20000</td>
<td>1.6</td>
</tr>
<tr>
<td>$\lambda_7$</td>
<td>10000</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 2: Qualifiers $q = (q_{\text{dist}}, q_{\text{qual}}, q_{\text{wo}})$ for all pairs of test functions.

<table>
<thead>
<tr>
<th>Failure rate</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\lambda_3$</th>
<th>$\lambda_4$</th>
<th>$\lambda_5$</th>
<th>$\lambda_6$</th>
<th>$\lambda_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>-0.00</td>
<td>0.21</td>
<td>0.41</td>
<td>0.52</td>
<td>0.78</td>
<td>0.84</td>
<td>1.00</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>-0.14</td>
<td>-0.49</td>
<td>-0.67</td>
<td>-0.00</td>
<td>0.00</td>
<td>-0.23</td>
<td>-0.56</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>0.45</td>
<td>0.90</td>
<td>0.43</td>
<td>0.52</td>
<td>0.00</td>
<td>-0.23</td>
<td>-0.56</td>
</tr>
<tr>
<td>$\lambda_4$</td>
<td>-0.07</td>
<td>0.15</td>
<td>0.23</td>
<td>0.28</td>
<td>0.56</td>
<td>0.73</td>
<td>0.82</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>-0.16</td>
<td>0.07</td>
<td>0.34</td>
<td>0.72</td>
<td>0.42</td>
<td>0.17</td>
<td>-0.15</td>
</tr>
<tr>
<td>$\lambda_6$</td>
<td>-0.29</td>
<td>0.52</td>
<td>1.00</td>
<td>0.46</td>
<td>0.21</td>
<td>0.04</td>
<td>-0.33</td>
</tr>
<tr>
<td>$\lambda_7$</td>
<td>-0.29</td>
<td>-0.30</td>
<td>-0.11</td>
<td>0.16</td>
<td>0.32</td>
<td>0.70</td>
<td>0.84</td>
</tr>
</tbody>
</table>

able to predict probabilistic functions or confidence bounds on $\lambda_X(t)$. Our choice was to predict the following function:

$$f_{\text{sim}}: X \subseteq \mathbb{R}^5 \rightarrow \mathbb{R} \times \mathbb{R} \quad (3)$$

$$f_{\text{sim}} \left( \begin{array}{c} t \\ \lambda_A(t) \\ q \end{array} \right) = [\lambda_X(t), \lambda_X(t)] \quad (4)$$

Two approaches to regress $f_{\text{sim}}$ are presented in this work: regression by neural networks and via Gaussian processes.

To evaluate the proposed methodology, a test set was created. We generated seven Weibull failure rates with different scale and shape parameters (Tab. 1):

$$\lambda(t) = \frac{\beta t^{\beta-1}}{\alpha^\beta} \delta(t) \quad (5)$$

To achieve a more realistic shape, we added a multiplicative noise term $\delta(t)$. Each combination of two functions $\lambda_i$, $\lambda_j$ was rated with all three qualifiers (see Tab. 2) which we also compromised with random noise. With these data, we constructed a set of training patterns of the form $(x; y)$, $x = (t, \lambda_i(t), q)$, $y = \lambda_j(t)$, sampling randomly over $t$ between 100 and 10000 operating hours.

3 Prediction

There are various different ways of tackling the imposed prediction problem. Trying to group possible approaches, we may divide into "black box" and "gray box" methods. The
former are very flexible, with a large set of parameters. If nothing or little about the underlying function is known, black box methods may be the right choice. The neural network regression that is investigated in subsection 3.1 belongs to this area. Gray box methods impose strong restrictions and premises on the underlying function. In exchange, they manage with a very small amount of parameters and learning processes may be reduced to simple optimization problems with exact solutions. A representative of this second group are Gaussian processes, which will be investigated in subsection 3.2.

3.1 Neural networks

Neural networks have proven their efficiency on regression problems over several decades and may still be considered as the tool of choice by many researchers [Cal99]. Thus, they seemed to be a feasible starting point for our research. We implemented a feed-forward neural network with one hidden layer (10 neurons) to regress $f_{sim}$. Unfortunately, neural networks are mainly used for the regression of deterministic functions which includes that learning methods do not account for uncertainty. To regress imprecise failure bounds from precise training data, we had to modify the output and the error criterion. The network contains three output neurons which predict the function $f_{NN}$ allowing the reconstruction of $f_{sim}$.

$$
\begin{align*}
\tilde{\lambda}_X(t) & 
\end{align*}
$$

The first output $\hat{\lambda}_X(t)$ tries to regress $\lambda_X(t)$. The second and third outputs, $\Delta(t)$ and $\Delta(t)$, are used to generate the prediction bounds (Fig. 2). In standard neural network training, the error vector between the prediction and the correct output vector is propagated through the network and the weights are fitted to minimize the error using backpropagation or gradient descent algorithms. This works fine for $\hat{\lambda}_X(t)$, but it is necessary to set up an error function for $\Delta(t)$ and $\Delta(t)$ depending on $\hat{\lambda}_X(t)$. Training patterns should not be found outside the bounds, but the bounds should also be tight. We found that a good trade-off was reached with:

$$
\begin{align*}
e(y_{NN}, \lambda_X(t)) &= 
\left(\frac{\lambda_X(t) - \hat{\lambda}_X(t)}{\Delta(t)} \times 10^{\text{sign}(\lambda_X(t) - \hat{\lambda}_X(t)) - \Delta(t)}\right) 
\end{align*}
$$

For weight optimization we used the resilient propagation method provided by MATLAB’s neural network toolbox.
3.2 Gaussian processes

Gaussian processes [SS02] are another feasible approach for the inference of \( f_{sim}(x) \). Gaussian processes assume that the training data is drawn from a model \( f_{sim}(x) + \epsilon \) where \( \epsilon \) is an additional random noise factor. Thus, prediction uncertainty is included in the model. The goal is now to infer \( y_{n+1} = f_{sim}(x_{n+1}) \) for an unseen input \( x_{n+1} \) as a probability distribution \( P(y_{n+1}|\vec{x}_{n}, \vec{y}_{n}, x_{n+1}) \) where \( \vec{x}_{n} \) and \( \vec{y}_{n} \) are the vectors of training data \((x_1,y_1), ..., (x_n,y_n)\). Another advantage of Gaussian processes for our purposes is that they specify a distribution directly over functions and thus the inference takes place directly in function space. For arbitrary failure rates this is certainly a plus. The basic idea in Gaussian processes is to interpret the training set as a realization of a stochastic process. 

**Gaussian process** Let \( f(x), x \in X \), be a stochastic process. Then \( f(x) \) is a Gaussian process if for any \( x_1, ..., x_n \in X, n \in \mathbb{N} \), the vector \( (f(x_1), ..., f(x_n)) \) is a multivariate Gaussian distribution.

A Gaussian process makes strong assumptions both about the noise and the interconnection between different function values which are not exactly fulfilled by our problem. Nevertheless the results seem to mimic reality quite good. Applying definition 3.2 to regression, the approach assumes that the joint probability distribution of any \( n \) output values, \( \vec{y}_n \), is an \( n \)-dimensional Gaussian distribution with covariance matrix \( C_n \):

\[
P(\vec{y}_n|\vec{x}_n, C_n) = \frac{1}{Z_n} \exp \left( -\frac{1}{2} \vec{y}_n^T C_n^{-1} \vec{y}_n \right)
\]

(8)

The construction of \( C_n \) and \( C_{n+1} \) hides the real learning process and will be explained later, while \( Z_n \) is a normalization constant. Now we want to predict \( y_{n+1} \) using our regression model and the new pattern \( x_{n+1} \). By applying Bayes’ law, we receive:

\[
P(y_{n+1}|\vec{x}_n, \vec{y}_n, x_{n+1}, C_{n+1}) = \frac{P(\vec{y}_n, y_{n+1}|\vec{x}_n, x_{n+1}, C_{n+1})}{P(\vec{y}_n|\vec{x}_n, C_n)} = \frac{P(\vec{y}_n, y_{n+1}|\vec{x}_n, x_{n+1}, C_{n+1})}{P(\vec{y}_n|\vec{x}_n, C_n)}
\]

(9)
The last expression holds because of the conditional independence between $y_n$ and $x_{n+1}$. By inserting eq. 8 into eq. 10, we get:

$$P(y_{n+1}|x_n, y_n, x_{n+1}, C_{n+1}) =$$

$$\frac{Z_n}{Z_{n+1}} \exp\left(-\frac{1}{2}y_{n+1}^T C_n^{-1} y_{n+1} + \frac{1}{2}y_n^T C_n^{-1} y_n\right)$$

By sampling over different values, a distribution for $y_{n+1}$ could be received. This results in a one-dimensional Gaussian distribution. A more efficient way is to compute the mean and variance of the prediction analytically. Introducing the following notation for $C_{n+1}$, mean $\mu$ and a variance $\sigma^2$ are given by [RBKTP05]:

$$C_{n+1} = \begin{pmatrix} C_n & k \\ k^T & \kappa \end{pmatrix}$$

$$\mu(x_{n+1}) = k^T C_n^{-1} y_n$$

$$\sigma(x_{n+1})^2 = \kappa - k^T C_n^{-1} k$$

We will use $\mu(x)$ and $\sigma(x)^2$ for computing the prediction intervals. Thus, the model is set up, but what are the parameters to fit in a learning process? The only question which arises is how to obtain the covariance matrix entries. This is done by a covariance function $f_C(x, x')$ proposed in [Mac97] returning the covariance between two training patterns $x, x'$:

$$f_C(x, x') = \theta_1 \exp\left(-\frac{1}{2} \sum_{i=1}^{I} \frac{(x_i - x'_i)^2}{r_i^2}\right) + \theta_2$$

where $I$ is the input dimension (in our case 5). $\theta_1, \theta_2, r_1...I$ are free parameters fixed by an optimization algorithm to minimize the model error. One nice property of this covariance function is that it places much more importance on data points close to the new pattern. $\theta_1$ and $\theta_2$ are overall scaling terms. Here, the grey box property can be observed. Compared to a neural network, the number of free parameters decreases enormously (in our example 7). We receive our interval boundaries from the 95% confidence intervals of the normal distribution which corresponds to $\mu(x) \pm 1.96 \sqrt{\sigma(x)^2}$ (Fig. 3). For the runs, we used the spider package [WEB05] for MATLAB.

4 Evaluation and Discussion

4.1 Evaluation

Both neural network and Gaussian process were trained on the set of test functions. To train neural networks, a large number of training patterns is beneficial as this may lead to an improved generalization capacity. Thus, each function pair was sampled over 100
different points in time resulting in 4900 training patterns. The Gaussian process generates its prediction directly from the training patterns and thus becomes incredibly slow (due to large matrix inversions) if the pattern numbers are high. Thus, the number of samples per function pair was kept low (30) resulting in 1470 training patterns. To validate the methodology, we applied a set of 1000 test patterns (10 function pairs, 100 samples) and measured the performance of both methods. We did not aim for exhaustive comparisons, as covariance function, network design and training method allow a large number of variations. Nevertheless, three properties of the result quality were measured: prediction imprecision (avg. distance between bounds), outlier percentage (correct value is outside prediction bounds) and average deviation of outliers from the predicted bound. All quantities were normalized according to $\lambda_X$. Imprecision and outliers are conflicting quantities. It is impossible to surpass an unknown level of imprecision caused by expert elicitation uncertainty without losing prediction quality. This is indicated by a large number of outliers. If the imprecision is too high, outliers converge to zero but the prediction becomes useless. The results can be seen in Tab. 3. Both approaches performed quite well for unknown test patterns. The neural network successfully managed to reduce the number of outliers to 12.9% without losing control of the prediction uncertainty. The Gaussian processes were also able to give useful prediction but showed a much more conservative way of uncertainty treatment. The number of outliers was 3.0%, on the other hand the imprecision value was much higher. Plots for an exemplary function pair are given in Fig. 5 and 4. The gray, dashed line indicates the failure rate of the known component, $\lambda_A$ which differed strongly from the failure rate to predict ($\lambda_X$: Validation). Nevertheless, it can be seen that both approaches managed to include the correct result almost completely in the prediction bounds. The neural network captured the form of $\lambda_X$ much better, leading to much more narrow confidence bounds. Nevertheless, both approaches are feasible ways to go, depending especially on the number of training patterns available.

4.2 Discussion

This paper has presented a novel way of reliability prediction. In-service data and expert estimates were combined to a single prediction method. Instead of a priori defining a prediction algorithm, we used methods from computational intelligence to regress the prediction function from training data. The method preserves elicitation uncertainty and thus allows robust reliability modeling. Due to the high availability of expert knowledge during the design process, the methodology is especially applicable in early design stages. In further work, practical applicability has to be proven using failure data (e.g. warranty figures) from real products. An application scenario could e.g. replace current FMEA sessions with similarity estimation, hopefully leading to more accurate predictions. Extensions may e.g. investigate more sophisticated expert elicitation procedures which include group decision models in a Bayesian [Coo91] or fuzzy framework. Aggregation methods to merge various failure rate estimates could improve scalability. But most promising, other prediction methods have to be investigated, first of all Bayesian neural networks (black box) and probabilistic support vector machines (grey box).
Table 3: Validation results of both approaches.

<table>
<thead>
<tr>
<th></th>
<th>Neural network</th>
<th>Gaussian process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outliers</td>
<td>12.9%</td>
<td>3.0%</td>
</tr>
<tr>
<td>Outlier distance</td>
<td>0.02073</td>
<td>0.0006351</td>
</tr>
<tr>
<td>Imprecision</td>
<td>1.005</td>
<td>6.544</td>
</tr>
</tbody>
</table>

Figure 4: Neural networks - Exemplary failure rate regression.

Figure 5: Gaussian process - Exemplary failure rate regression.
Acknowledgements

The authors would like to thank Irman Hakim, Thomas Hensel and Jörg Petersen for helpful ideas and discussions.

References


