A FRAMEWORK FOR ASSISTING LEARNERS BY INCORPORATING KNOWLEDGE TO AID IN PREDICTING NERVE GUIDANCE CONDUIT PERFORMANCE

by

William Frederick Koch III

A THESIS

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE - COMPUTER ENGINEERING

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2013

A FRAMEWORK FOR ASSISTING LEARNERS BY INCORPORATING KNOWLEDGE TO AID IN PREDICTING NERVE GUIDANCE CONDUIT PERFORMANCE

ABSTRACT

Collecting an adequate number of samples to make accurate predictions can be challenging in many domains. Events may occur infrequently or sampling techniques may be costly to perform to collected the desired number of samples. The resulting collection may be sparse, and/or incomplete which can cause inductive machine learning algorithms to perform unpredictably. Typically there exists additional knowledge from domain experts which can be used to produce more accurate predictions. This thesis presents a method for fusing prior or post domain knowledge into information provided by a machine learning system while acting independently of the learners training algorithm. The influential strength of the additional knowledge is dependent on the machine learners confidence of the prediction. The presented method is applied to a real-world application to predict the performance of various nerve guidance conduits (NGCs) where data is limited because a single experiment can take over a year to complete. The ability to predict NGC performance will provide for a better understanding and insight for theorizing successful strategies for NGC development. A benchmark demonstrates how the presented model provides substantial increases in accuracy in predicting the performance of a NGC over a single neural network, bootstrap aggregated neural network, and previous prediction attempts by the SWarm Intelligence based Reinforcement Learning (SWIRL) system.

Author: William Frederick Koch III

Advisor: Yan Meng

Date: July 1, 2013

Department: Computer Engineering

Degree: Master of Science - Computer Engineering

Dedication

This thesis is dedicated to my Mom and Dad who have fully supported me through my life and academic career. I am very grateful for all of the sacrifices you have made for me and the help you have given to me over the years. I would also like to dedicate this thesis to my brothers Coleman, Spencer and Carter. I am so proud of you guys, you are all extremely gifted and have so much to offer this world.

I love you all very much.

Acknowledgments

I would like to acknowledge my advisor Professor Dr. Yan Meng who since the beginning of my graduate studies has had nothing but support and confidence in my studies. Everything I have learned about academic research and writing is attributed to her. As most of my research was focused in bioinformatics I express my gratitude to the biological perspective Professor Dr. Xiaojun Yu has given me. Additionally I would like to thank Wei Chang for his efforts in compiling data used in this thesis. Finally I would like to thank my colleague Munish Shah who has taught me everything I know about nerve guidance conduits, the focus of this research. The majority of data collection, preparation and interpretation is accredit to him. This work would not be possible without Munish's dedication and passion on the subject. It was a pleasure working with everyone on this research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Assembling a balanced training set of sufficient size is critical for a learner to make accurate predictions. As the number of observations increase a better understanding of the true underlying target distribution is uncovered and prediction accuracy can increase.

Increasing the sample size, however, is only beneficial if there is a low correlation between samples. Furthermore there are many cases in which the number of samples available are limited and the required sample size can not be reached for the desired accuracy. Incomplete and unbalanced training sets can lead to high uncertainty in the predictors output [26]. Typically there exists experts in the target domain which are knowledgeable of the observed system. Incorporating their domain knowledge can help fill the gaps where data is insufficient and improve accuracy in prediction. This is of particular importance in modelling systems where data collected is obtained from long-term experiments. Long-term experiments occur in many domains including but not limited to biology, agriculture, and astronomy. A 20 year experiment to study genomic evolution through E. coli was conducted by [7]. Long-term agroecosystem experiments (LTAE) are conducted to predict global agricultural sustainability. LTAEs can occur for more 50 years and are referred to as classical [65]. Quasi-cycles from cosmic X-rays, which can occur for many years, are difficult to observe due to equipment limitations [63]. The limitation in data makes predicting these cycles challenging.

Additionally there are situations in which new knowledge is discovered by system experts after models have been created and distributed to their target environment. This leads us to the following scenario; suppose we encounter a task that is unfamiliar to us. Due to our limited knowledge our confidence is low in regards to performing it successfully. However if we are supervised by a trusted source available to assist us with additional knowledge on how to perform the task, we are given the ability to adjust our initial approach to form a more accurate result.

This scenario has led to the creation of a framework for Assisting Learners by INcorporating Knowledge (ALINK) [46]. ALINK provides flexibility and robustness allowing for incorporation of prior or post domain knowledge, in a non-invasive fashion, that is the fusion of knowledge from the expert system and learner occurs outside of the training process which allows it to be used with any inductive learning algorithm. ALINK utilizes the confidence of the learner to determine the degree of influence the additional knowledge will have on the prediction. When the learners confidence is low the learner gravitates toward the information provided by the available knowledge. If the learners confidence is high, it will be less interested in the additional knowledge and trust its own ability to produce an accurate prediction.

ALINK serves as a guideline for fusing domain knowledge with that of a learner. Many implementations may be used. Knowledge may be represented as strict or fuzzy truth and may be in the form of constraints, bounds or approximations for all or selective input parameters. All available knowledge is used to nudge the prediction output in the correct direction when the learner is uncertain. The prediction may come from any learned system such as a neural network (NN), Support Vector Machine (SVM), Bayesian network, etc. Furthermore any method for determining the prediction confidence may used such as the delta method, Bayesian framework, mean-variance estimation (MVE) or by bootstrapping which will all be discussed in Section 3.2.

The remainder of this thesis is outlined as follows. In Chapter 2 a literature review is conducted surveying methods for incorporating domain knowledge into inductive machine learning algorithms. Chapter 3 describes in detail the frameworks approach to incorporating domain knowledge. An overview of how the confidence of a learner is determined is also discussed in this chapter summarizing popular methods with an emphasis on neural network implementation. Additionally the implementation of ALINK used in experimentation is discussed in detail here. A demonstration of the accuracy of the ALINK in a real-world application predicting the performance of nerve guidance conduits in presented in Chapter 4. Chapter 5 summarizes experimental results and in Chapter 6 future work is discussed.

Chapter 2
Literature Review

2.1 Domain Knowledge Integration

Integrating prior domain knowledge can increase prediction accuracy when observations used for training are limited. Domain experts have learned about the target system and hold vital information that can be used to aid in the prediction of the system. The method in which data is extracted from the experts in the form of knowledge and integrated into a machine learning algorithm is the challenge. Additionally knowledge about the system can be used to generate synthetic observations to aid in training. Various methods for incorporating prior domain knowledge have been surveyed by [55] and [78]. Two popular methods are through virtual examples and hints.

2.1.1 Virtual Examples

Prior domain knowledge can be used to generate new training examples to be used in the learning process. These new examples are known as virtual examples [55]. This is particularly useful when the training data is of insufficient size. Virtual examples can be generated by synthesizing entirely new examples [61] or by transforming the original data [54, 55].

Synthetic Examples

Synthetic training examples are training examples fabricated from domain knowledge. Since they are unique they have the ability to have low correlation to the observed data they can aid in filling in gaps in training data.

The ALVINN (Autonomous Land Vehicle In a Neural Network) project [61, 62] utilizes a NN to process video data to pilot a modified Chevy van. Previously, heavy preprocessing has prevented successful real time training using vision sensory data [48]. To overcome this limitation ALVINNs original training scheme used back-propagation (BP) taught on synthetic road scenes as input with its corresponding driving controls as output. Although ALVINN was successful at navigating a short distance at low speeds under various weather conditions, using synthetic examples had its shortcomings. Generating the synthetic examples was very computationally expensive and when tested using real sensory data caused poor performance.

Recognition of a material can allow for a robotic system to interact with the environment more intelligently. [50] has created virtual materials to form a new virtual database MPI-VIPS (Virtual texture under varying Illumination, Pose and Scales). Using shaders commercially available and from the graphics community they were able to use Autodesk 3ds Max to rebuild scenes from the original materials database. Using automated scripting scene changes could be done automatically rather than manually in a real world setting. Some virtual materials with complex light interactions did however differ from there real world counterparts.

Invariance Transformation

Invariance transformation [78] involves a piece of information P to undergo a transformation s where the target function result is the same as the original, that is,

$$f(s(P)) = f(P) \tag{2.1}$$

For example say we are training a network to recognize an image of a cat. If only trained on images of cat faces the network will only be able to recognize a cat by its face. If we use a rotation transformation and capture images of a cat from different angles it will allow the network to better generalize what a cat looks like. Possible geometrical transformations of image data include shearing, scaling, rotation and distortion [54, 70]. Virtual examples by transforming the original data set are useful in pattern recognition problems such as vision and speech.

Invariance transformations however are limited to data transformation which satisfies the initially defined condition (2.1), additionally a challenge involved in this approach is determining appropriate methods for creating novel transformations to overcome the inherit high correlation to the original information. [78] has proposed methods using a linear class technique to create novel 2-D virtual views of 3-D objects without knowing what the true 3-D object looks like.

Virtual examples have the main advantage of being able to be applied to any inductive learning system and to the learner the virtual examples are identical to the observed examples. Therefore virtual examples are not limited to any particular training algorithm. Once the virtual examples are created they are viewed no differently than any other training example from the original data set. This is particularly useful when using 3rd party software such as Weka [28], Encog Machine Learning Framework, and MATLAB's Neural Network Toolbox.

2.1.2 Hints

Hints are defined as additional pieces of information about the target function to aid in the learning process [1]. They provide complexity and informational value [2]. The complexity value allows for the number of steps to estimate the target function to be reduced while the information value provides means to reduce the search space.

The target function f is estimated by some hypothesis g. They error describing the agreement between the target and approximation is defined as $E(g, f)$. Below are a list of popular hints as discussed in [1] and [78]:

Invariance Hint

The invariance hint asserts that $f(x) = f(x')$ for a pair x and x'. The input x under goes some sort of transformation to produce x'. Although the input under goes this transformation the targets are still identical. For example in image recognition, if an original image x is skewed or scaled to produce x' it is still the same image. The error produced by a single example is defined by (2.2). The error is derived from the hypothesis of the original input and the transformed input. The target function does not need to be known as the error is represented as an assertion between the two outputs. Invariance hints however leave it to the user to determine how the input can be transformed and to what degree. As the transformed input x' is derived from the original input it can lead to high correlation between the two [78]. Others such as [55] have focused solely on the creation of this transformation.

$$e_m = [g(x) - g(x')]^2 \tag{2.2}$$

Monotonicity Hint

The monotonicity hint asserts that $f(x) \geq f(x')$ for a pair x and x', that is for a particular x' it is increasing fulfilling the $x \leq x'$ property. The error associated with this hint is defined by (2.3).

$$e_m = \begin{cases} [g(x) - g(x')]^2 & \text{if g(x) > g(x')} \\ 0 & \text{if g(x) } \leq \text{ g(x')} \end{cases} \tag{2.3}$$

It was shown in [66] that monotonicity hints can be applied to credit card application task and in medical diagnosis.

Example Hint

Typical training examples can also be represented as example hints. Examples hints assert that given $[x_1, f(x_1)], ..., [x_N, f(x_N)]$ the function f is the correct value for the inputs $x_1, x_2, ..., x_N$. The error e_0 (2.4) is denoted by the subscript 0 to represent an example hint. Example hints are used to transparently treat observed training examples the same as other hints.

$$e_0 = [g(x_n) - f(x_n)]^2 \tag{2.4}$$

Approximation Hint

Approximation hints assert that for $x \in X$ then $f(x) \in [a_x, b_x]$. For input x, the value of f is only approximately known. The error is defined by (2.5).

$$e_m = \begin{cases} [g(x) - a_x]^2 & \text{if g(x)} < a_x \\ [g(x) - b_x]^2 & \text{if g(x)} < b_x \\ 0 & \text{if g(x)} \in [a_x, b_x] \end{cases} \tag{2.5}$$

Catalytic Hint

Up until this point the previous discussed hints have been incorporated into the learning processes by representing the hints as virtual examples. Catalytic hints proposed by [68] are incorporated quite differently. During training an additional output node is added to the network which is learned simultaneously to the target output node. Once the network is trained, the catalyst node is removed from the network. Typically the output of this catalyst node is used to emphasize something about the target output. Designing this hint to produce optimal results can be challenging. It has been shown by [41] that fuzzy knowledge can be used as a catalyst function.

In order for hints to be used in the learning process they must be represented in a way the learner can interpret. Hints can be represented as virtual examples or duplicate examples. The virtual examples previously discussed are essential the same to duplicate example in [2] however hints represented as virtual examples are quite different.

The total hint error E (2.6) is a combination of the errors of all the M hints.

$$E = \hat{E}(E_0, E_1, E_2, ..., E_M) \tag{2.6}$$

For hints used to train a NN, backpropagation can be used in which the gradient is defined by (2.7). Here w is the adjusted weight.

$$\frac{\partial \hat{E}}{\partial w} = \sum_{m=0}^{M} \frac{\partial \hat{E}}{\partial E_m} \frac{\partial E_m}{\partial w} \tag{2.7}$$

Virtual examples created from hints have two advantages over duplicate examples [1]. First, since training on virtual examples is independent of training on the observed data set, more control is available on the level of effect the hints will have in the learning process. Secondly since a target output is not required for many hints it allows for an infinite number of virtual examples to be created.

Chapter 3
Model

Motivation to create such model was a result of working with data reporting performance of various nerve guidance conduit (NGC) construction methods. This research which will be discussed in greater detail in Chapter 4. The difficultly in deriving a proper model to predict NGC performance was due to a limited data set and its high dimensionality. When a feed-forward neural network (FFNN) was trained on this data set it was quite inaccurate. There was however knowledge rules to accompany the data. It was known that this additional information needed to be integrated in order to provide for a more accurate prediction.

The primary disadvantage to using hints is the method in which they are integrated into the learning processes. As discussed in Section 2.1.2 the popular method for incorporation was modifying decent algorithms. In order for other training algorithms to be used they must also be modified to take into account the error of the hints. As the use of hints is not standard practise it prohibits the use of out of the box training algorithms in 3rd party machine learning toolboxes. Duplicate examples, (or virtual examples referred to by [55]), provide for transparency between synthetic generated examples and those collected examples by observation allowing for the use in any 3rd party machine learning toolbox. However if the generated examples have a high correlation to the original data set this is going to do little to help the models generalization.

Reviewing the original FFNN output predictions of NGC performance, the network had higher prediction accuracy in areas where data was balanced, however where gaps in the data occurred accuracy dropped. This lead to the question; how could the learners confidence be determined so when needed additional knowledge could be used to help the learners ability to make a prediction?

In the beginning stages of predicting NGC performance the knowledge rules that existed were relatively phrased to a control. They could tell us if based on a selection of parameters if it would perform better or worse than a control however they could not give us absolute values of the actual performance. For this reason a method was created to try and influence a learner in the same relative manner.

The original idea was to create dynamic neurons in a NN. The neurons firing sensitivity would be a function of confidence and additional knowledge. The change in sensitivity was an attempt to influence the model toward the correct prediction relative to its normal firing sensitivity. An increase in sensitivity provided for a higher output, while decrease in sensitivity provided a lower output value. An increase and decrease in sensitivity can be seen with a hyperbolic tangent function ϕ in Figure 3.1.

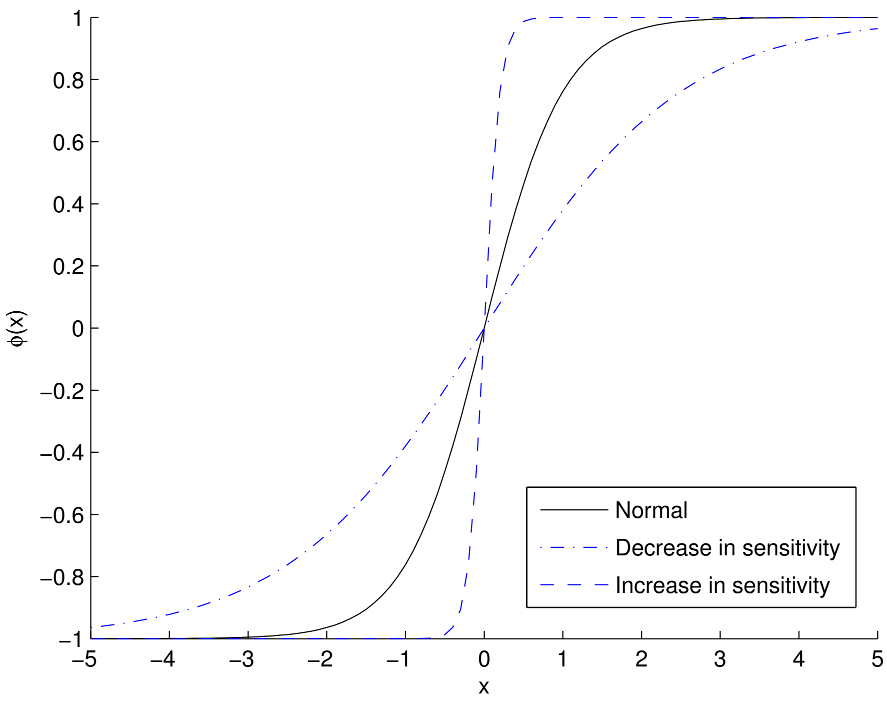

Figure 3.1: Change in neuron sensitivity of a hyperbolic tangent function. An increase in sensitivity causes for higher output while decrease in sensitivity results in lower output.

The model displayed in Figure 3.2 is comprised of a fuzzy system (FS) to represent the knowledge, confidence and a NN. The output of the fuzzy system I is a measure of influence the knowledge will have to adjust the sensitivity of the neurons. The confidence C is determined by calculating the distribution of input occurrence during training and evaluating the current input parameters during testing against this distribution. A degree of influence I°, a function of the confidence and knowledge, provided the measure to change the neuron sensitivity.

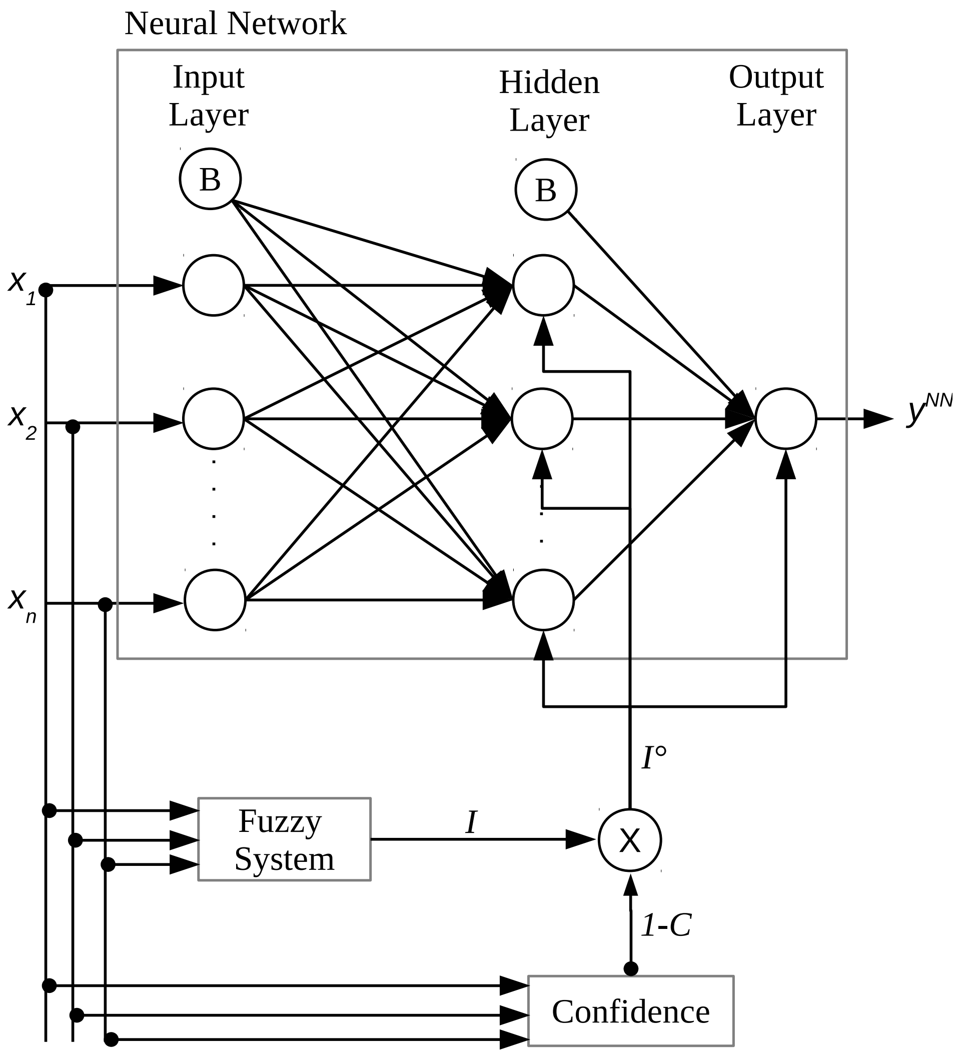

Figure 3.2: Original method for attempting to incorporate knowledge into a learner by level of confidence. This method changes the firing sensitivity of the neurons to influence its ability to make a prediction.

Although the idea of dynamically adjusting neuron sensitivity may serve beneficial in other implementation it resulted in in little to no improvement in this model. Furthermore the inability to adjust each neuron independently did not allow for the flexibility needed. The output difference in the neural network and fuzzy system added additional complexity to integration. Additionally the construction method for determining confidence was inaccurate. The confidence was more of a indication of data distribution rather than confidence of the learner which was required. A method for influencing a learners decision with additional knowledge based on the learners confidence however seemed promising thus further investigation lead to the birth of a framework for Assisting Learners by INcorporating Knowledge (ALINK).

3.1 Overview

The approach used by the ALINK framework is to fuse the information provided from the additional knowledge and the learner to provide for the most accurate prediction. It is assumed that there is a direct correlation between the learners confidence and the impact of the additional knowledge should have on the output. We want to provide assistance to the learner when its confidence is low in the form of additional knowledge we know about the target function. The domain knowledge attempts to fill the void in cases where the learner is not confident. The learner output is adjusted depending on how confident it is on estimating the output for the current input values. When the learners confidence is low we want the learners output to gravitate toward the information provided by the domain knowledge. If the confidence is high, we the learner output is trusted more and rely on information provided from domain knowledge less. We do not want to trust the knowledge fully as the knowledge provided may be incomplete or fuzzy.

Furthermore the knowledge integration methods discussed in Section 2.1 are all for incorporating prior domain knowledge. ALINK allows for post knowledge incorporation. In events that additional knowledge is discovered after a learner has been trained, additional knowledge may be added without the learner having to be retrained. Depending on network complexity, training algorithm, size of data set and computer hardware, a learners training time will vary. For machine learning algorithms deployed on limited hardware it may be unrealistic for them to be able to perform their own training. However if the new data is in the form of knowledge, the knowledge only need to be replaced in ALINK adding no additional complexity.

In order for ALINK to correctly fuse the information of the provided knowledge and prediction of the learner, a method for measuring the learners prediction confidence must be used. ALINK uses prediction intervals to provide this measure. Many different methods for computing prediction intervals exist, the appropriate one should be chosen for the application.

3.2 Prediction Interval Construction Methods

The prediction interval (PI) identifies the likelihood of an observation falling within a particular range with respect to a predetermined $(1-\alpha)\%$ confidence interval. The PI serves as a measure of accuracy, indicating how well the prediction can be trusted. Wide PIs express a higher level of uncertainty thus implying the output is less trust worthy, while narrow PIs suggest there is a smaller range the observation will fall and the prediction is more trust worthy.

Machine learning algorithms may react unpredictably when there is uncertainty in the input. This uncertainty may be a result of the training data being sparse or incomplete. Providing an indication of the level of accuracy of the prediction is particularly important in black box models such as neural networks.

There are two components to consider when determining the confidence of a models estimation, the confidence interval and prediction interval [64]. Targets of a system are typically modelled by (3.1),

$$t_i = y_i + \epsilon_i \tag{3.1}$$

where t_i is the measured target for the ith sample, y_i is the true output and ϵ_i is the noise introduced during measurement. As we typically do not know the true function, we attempt to estimate the function output with $\hat{y}_i$. Introducing the estimate to (3.1) results in the models confidence (3.2).

$$t_i - \hat{y}_i = [y_i - \hat{y}_i] + \epsilon_i \tag{3.2}$$

The left hand side defines how far the estimate is from the target which is known as the PI. The difference between the true function and estimated function $y_i - \hat{y}_i$ is defined as the confidence interval.

The PI is the uncertainty in the estimation. It is based primarily on the model variance. The model variance (3.3) is found by the variance in model parameters $\sigma^2_{\hat{y}_i}$ and the variance in noise $\sigma^2_{\epsilon_i}$.

$$\sigma^2 = \sigma^2_{\hat{y}_i} + \sigma^2_{\epsilon_i} \tag{3.3}$$

Many techniques have been developed for constructing PIs. Some of the popular methods including the delta method, Bayesian framework, mean-variance estimation (MVE), and bootstrapping.

3.2.1 Delta Method

The delta method is a common method to find the variance of a function of a random variable, specifically for complex non-linear functions. The method seeks to find a simpler version of the underlying function by computing a linear approximation. The variance is then found from the simpler linear functions. The linear approximation is usually found by Taylor series expansion about the mean [56]. The PI can be found by treating the model parameters of a function approximation as random variables.

It has been show by [19, 23, 35] that the delta method can be applied to form

PIs for for neural networks. Each neural network is treated as a nonlinear regression model. Finding the linear function involves computing the partial derivatives of the NN model with respect to the model parameters in the form of a Jacobian matrix. The variance in model parameters is defined as

$$\sigma_{\hat{y}}^2 = \sigma_\epsilon^2 g^T (F^T F)^{-1} g \tag{3.4}$$

where g^T is neural network output gradient, F is the Jacobian matrix (3.5) and σ_ϵ^2 is the variance in noise.

$$F_{i,j} = \frac{\partial f(x_n, \hat{w})}{\partial \hat{w}_p} \tag{3.5}$$

Each element in the Jacobian matrix is a partial derivative of the model function $f(x_n, \hat{w})$ with respect to the weight $\hat{w}_p$. If we replace $\sigma_{\hat{y}_i}^2$ in (3.3) with (3.4) we derive the total variance for the delta method by,

$$\sigma_0^2 = \sigma_\epsilon^2 (1 + g_0^T (F^T F)^{-1} g_0) \tag{3.6}$$

From (3.6) the $(1-\alpha)\%$ confidence is defined as,

$$\hat{y} \pm t_{n-p}^{1-\frac{\alpha}{2}} s_\epsilon \sqrt{1 + g_0^T (F^T F)^{-1} g_0} \tag{3.7}$$

where $t_{n-p}^{1-\frac{\alpha}{2}}$ is the cumulative t-distribution with $n-p$ degrees of freedom and s_ϵ is the unbiased estimate of σ_ϵ^2. During construction, deriving the Jacobian matrix can be computationally expensive and should be preformed offline [45].

3.2.2 Bayesian Framework

In Bayesian statistics, conclusions are draw from not only the sampled data but also from prior events. A PI is derived from the posterior distribution. Initially we have a prior probability of a random variable $p(\theta)$. The likelihood of observation X occurring is $p(X|\theta)$. The posterior probability is then defined by (3.8).

$$P(\theta|X) = \frac{P(\theta)P(X|\theta)}{P(X)} \tag{3.8}$$

Once the expected value and variance are computed from the distribution the prediction interval can be derived. [23, 45] has shown how the Bayesian framework can be applied to producing PIs for a neural network.

Using Bayesian training, weights w in a neural network M can be updated based on the current training data D according to,

$$P(w|D, \rho, \beta, M) = \frac{P(D|w, \beta, M)P(w|\rho, M)}{P(D|\rho, \beta, M)} \tag{3.9}$$

where ρ and β are hyperparameters of the cost function, that is they are parameters of the prior distribution. The probability of a particular data example occurring given a set of particular weights is represented by $P(D|w, \beta, M)$, assuming errors occurring are normally distributed is defined as,

$$P(D|w, \beta, M) = \frac{1}{Z_D(\beta)} e^{-\beta E_D} \tag{3.10}$$

Here $Z_D(\beta) = \left(\frac{\pi}{\beta}\right)^{\frac{n}{2}}$ where n is the number of training examples and E_D is sum of squared errors (SSE). The distribution of the weights given the neural network is defined by $P(w|\rho, M)$, in which it is also assumed errors normally distributed,

$$P(w|\rho, M) = \frac{1}{Z_w(\rho)} e^{-\rho E_w} \tag{3.11}$$

Here $Z_w(\rho) = \left(\frac{\pi}{\rho}\right)^{\frac{p}{2}}$ where p is the number of NN parameters and the sum of squares of the NN weights is E_w. The term $P(D|\rho, \beta, M)$ in the denominator is used for normalization to ensure a probability of one. With the assumption of normally distributed errors (3.10) and (3.11) are substituted into (3.9),

$$P(w|D, \rho, \beta, M) = \frac{1}{Z_F(\beta, \rho)} e^{-(\rho E_w + \beta E_D)} \tag{3.12}$$

$$H^{MP} = \rho \nabla^2 E_w + \beta \nabla^2 E_D \tag{3.13}$$

Substituting the noise in the training examples σ_D^2 for $\sigma_{\epsilon_i}^2$ in (3.3) and the variance in neural network parameters $\sigma_{w^{MP}}^2$ for $\sigma_{\hat{y}_i}^2$ the total variance in the model is defined as,

$$\sigma_i^2 = \sigma_D^2 + \sigma_{w^{MP}}^2 \tag{3.14}$$

The variance is further defined to,

$$\sigma_i^2 = \frac{1}{\beta} + \nabla_{w^{MP}}^T \hat{y}_i \left(H^{MP}\right)^{-1} \nabla_{w^{MP}} \hat{y}_i \tag{3.15}$$

where $\nabla_{w^{MP}}^T$ is the output gradient per the most probable weights w^{MP} and H^{MP} is the Hessian matrix of the training cost function. Based on the total variance the $(1-\alpha)\%$ PI is then defined as,

$$\hat{y}_i \pm z^{1-\frac{\alpha}{2}} \left(\frac{1}{\beta} + \nabla_{w^{MP}}^T \hat{y}_i \left(H^{MP}\right)^{-1} \nabla_{w^{MP}} \hat{y}_i\right)^{\frac{1}{2}} \tag{3.16}$$

where $z^{1-\frac{\alpha}{2}}$ is the quantile function of a normal distribution. Like the delta method the Bayesian method is computationally expensive during construction because of the Hessian matrix.

3.2.3 Mean-Variance Estimation (MVE)

Mean-Variance Estimation (MVE), proposed by [53], assumes that errors are normally distributed about the true ideal mean, therefore if the mean and variance is identified the PI can be constructed. Based on the fact that the output is a function of the input, the error must also be a function of the input. A second learning function is used to estimate the variance by being taught on the noise observed in the data set. The two learners operate in parallel, both fed the same input data, one learner producing the estimated output, and the second learner providing the estimated variance in the output.

[53] used neural networks as the learner to derive the PI. Unlike the delta and Bayesian methods, MVE PI are dynamic because of the use of the second neural network NN_σ computing the variance. In order to train NN_σ there is a second training data set. This data set is comprised of the same input values used to train NN_y however the variance with respect to the training data outputs must be calculated for the NN_σ data set target output. During training NN_σ is then minimized by the following cost function,

$$C = \frac{1}{2}\sum_{i=1}^{n}\left[ln(\hat{\sigma}_i^2) + \frac{(t_i - \hat{y}_i)^2}{\hat{\sigma}_i^2}\right] \tag{3.17}$$

With both NN_y and NN_σ trained the $(1-\alpha)\%$ PI can then be calculated by,

$$\hat{y}(x, w_y) \pm z_{1-\frac{\alpha}{2}}\sqrt{\hat{\sigma}^2(x, w_\sigma)} \tag{3.18}$$

where $\hat{y}(x, w_y)$ is the predicted output from neural network NN_y, $\hat{\sigma}^2(x, w_\sigma)$ is the output NN_σ representing the variance and $z_{1-\frac{\alpha}{2}}$ is the standard normal distribution value. The notation w_y and w_σ represent the weights for NN_y and NN_σ respectively.

The absence of derivative calculations provide for a simpler model than the delta and Bayesian methods however its generalization is poor as it assumes that NN_y will compute the true mean which is problematical in real-world applications [45].

3.2.4 Bootstrapping

Bootstrapping [24] is a re-sampling technique based on the idea that the true probability distribution of a population can be modelled by re-sampling an empirical distribution obtained from the original observed data set.

There are two bootstrap sampling approaches for regression, sampling by pairs or residuals [69]. If the observed training data set is of size n, the bootstrap pairs approach follows that a collection of n examples,

$$\{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), ..., (\vec{x}_n, \vec{y}_n)\}$$

is drawn with replacement from the empirical distribution. This forms the bootstrap sample

$$\big\{(\vec{x}_1^{*b}, \vec{y}_1^{*b}), (\vec{x}_2^{*b}, \vec{y}_2^{*b}), ..., (\vec{x}_n^{*b}, \vec{y}_n^{*b})\big\}$$

denoted by an asterisk $(*)$ for the *bth* sample. The residual approach samples the model residuals $r_i = y_i - \hat{y}_i$ as opposed to the training observations used in the bootstrap pairs approach. Each sampled residual is added to the predictor $\hat{y}$ to form the bootstrap sample

$$\big\{(\vec{x}_1, \hat{y_1} + r_1^{*b}), (\vec{x}_2, \hat{y_2} + r_2^{*b}), ..., (\vec{x}_n, \hat{y_n} + r_n^{*b})\big\}$$

where $\hat{y}_i = y(\vec{x}_n; \hat{\theta})$ and $\hat{\theta}$ are the estimated model parameters. The primary difference between the two methods as described by [8] is that the pairs approach provides an unconditional bootstrap distribution while the residual approach provides a conditional bootstrap distribution. The conditional factor retains to the observed data set. The re-samples of the training observations in the pairs approach allow for an underlying distribution of the population to be found rather than the distribution found strictly from the training observations used by the residuals approach. One must select the appropriate method pertaining to their target objective.

A bootstrap aggregated (bagged) predictor is an ensemble of predictors trained on unique bootstrap samples and combine to form an aggregated predictor [10].

The bagged predictor is constructed by creating B predictors. B bootstrap samples are then created from the original training data set. The *bth* predictor is trained on the *bth* bootstrap sample. Result $\hat{y}$ from a bagged predictor is found for regression tasks by computing the mean (3.19) of each individual predictors outputs $\hat{y}^b$ while in classification a vote is taken.

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b \tag{3.19}$$

The model variance resulting from errors in model parameters is calculated by the unbiased sample variance of the output of all learners in the ensemble,

$$\sigma_{\hat{y}}^2 = \frac{1}{B-1}\sum_{b=1}^{B} (\hat{y}^b - \hat{y})^2 \tag{3.20}$$

As a result of sampling with replacement, when a bootstrap sample is created, about 37% of the training examples are omitted, these are referred to as out-of-bag (OOB) [11]. The noise term $\sigma_{\epsilon_i}^2$ in (3.3) is estimated by using a second bagged NN which is trained on a new data set $\{(x_i, r_i^2)\}_{i=1}^{n}$ which is constructed from OOB data.

Here x_i are the inputs from the OOB samples and r_i^2 are the variance squared residuals found by (3.21). The variance squared residual is a function of the target value of the OOB sample t_i, the ensemble output mean $\hat{y}_i$ and variance $\sigma_{\hat{y}_i}^2$ between all network outputs. $\hat{y}_i$ and $\sigma_{\hat{y}_i}^2$ are found by (3.19) and (3.20) respectively. Training the noise estimator on OOB samples provides for an unbiased estimation.

$$r_i^2 = max((t_i - \hat{y}_i)^2 - \sigma_{\hat{y}_i}^2, 0) \tag{3.21}$$

The network estimating $\sigma_{\epsilon_i}^2$ is trained until the log likelihood error function (3.22) is minimized.

$$L \equiv -\sum_v log\left[\frac{1}{\sqrt{2\pi\chi^2(\vec{x})}} exp(-\frac{r^2(\vec{x})}{2\chi^2(\vec{x})})\right] \tag{3.22}$$

The standard deviation of the total variance is then scaled by the t-score which can be found in a Student's t-table or as [31] has suggested by ensuring no more than $100\alpha\%$ of the training cases statisfy $t_{\alpha/2[v]} \leq |t(x) - \hat{y}|/s(x)$. The final prediction interval is defined by (3.23).

$$\hat{y} \pm t_{\alpha/2[v]}\sqrt{\sigma_{\hat{y}}^2 + \sigma_{\epsilon}^2} \tag{3.23}$$

3.3 Architecture

ALINK has been developed to provide guidelines for producing the desired effects as discussed prior. ALINK is comprised of four components displayed in Figure 3.3. The learner component is trained on observed data and produces an estimate of the target function. The implementer can determine which inductive learning algorithm suits their needs best whether a NN, SVM, etc. The confidence component computes the learners PI for the current inputs which can be any method discussed in Section

3.2. The knowledge component provides a separate estimation of the target function based on acquired domain knowledge. This typically would be represented as an expert system. The supervisor component determines the output of the system based on information provided from the learner, knowledge, and confidence components.

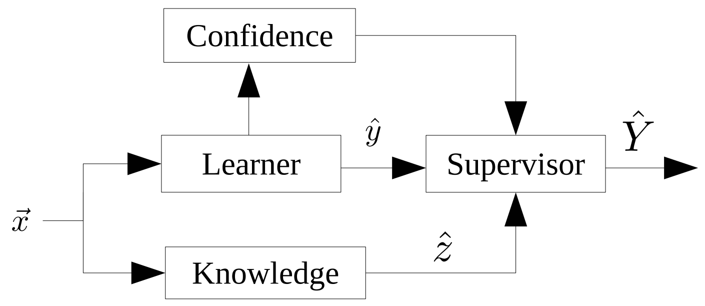

Figure 3.3: Overview of ALINK components. Current input values $\vec{x}$ are fed to the learner and knowledge base. Both produce estimates of the target output. How the estimates are fused together to form the output is determined by the prediction confidence.

ALINK is flexible enough to use any inductive machine learning algorithm as long as a prediction interval can be derived. Additionally the knowledge component is not restricted to any particular implementation just as long as domain knowledge can be used to produce its own prediction. The supervisor component fuses together $\hat{y}$ and $\hat{z}$ based on the learners confidence and is the novel work presented here.

Using the output from the knowledge component $\hat{z}$ we derive a more specific PI estimate $\widehat{PI}$ from (3.24) based on which side of the PI the knowledge estimate lies. We assume the true output is more likely to reside in the direction of the result from the knowledge estimate. We use the notation $PI_{\uparrow}$ to represent the upper PI and $PI_{\downarrow}$ as the lower PI. In methods where the PIs are symmetric $\widehat{PI}$ will simply equal $PI/2$.

$$\widehat{PI} = \begin{cases} PI_{\uparrow} & \hat{y} < \hat{z} \\ 0 & \hat{y} = \hat{z} \\ PI_{\downarrow} & \hat{z} < \hat{y} \end{cases} \tag{3.24}$$

There must be a way to determine the level of agreement of the knowledge estimate and learner. Since the learner will gravitate toward the knowledge estimate we need to know to what degree. The gravitational attraction $g(\hat{y}, \hat{z})$ will provide this measure. As a proof of concept g is implemented as the linear relationship of $\hat{z}$ and $\hat{y}$, where m is the slope,

$$g(\hat{y}, \hat{z}) = m\hat{z} - \hat{y} \tag{3.25}$$

Different methods may be used, however since the additional knowledge is to fill gaps in training data the gravitational pull will typically be in some form of an increasing function. For instance the designer may not want the relationship between the knowledge estimate and learner to be linear. Perhaps if there is a large disagreement between the two the $\hat{z}$ should have less influential power the father way it is or vice versa. Other functions such as polynomials and exponentials may be better applicable.

The supervisor adjusts the learners output according to (3.26) to produce the final output $\hat{Y}$ where $range(y)$ is the range in which the output y may represent. The ratio of $\widehat{PI}$ and $range(y)$ provide for a scaled representation of the learners confidence, that is, $\widehat{PI}$ is normalized by the total possible range of y.

$$\hat{Y} = \hat{y} + g(\hat{y}, \hat{z})\frac{\widehat{PI}}{range(y)} \tag{3.26}$$

If both estimations are in agreement then $g = 0$ which cancels the second term in (3.26) and ALINK output simply becomes the output of the learner.

ALINK will use all available knowledge, that being said there may not be knowledge available about every single parameter. In high dimensional problems this may be more of a concern. Essentially ALINK evaluates the given inputs and compares the input values to the knowledge base to determine if knowledge exists for any of the expressed input values. This process is outlined in the flow chart in Figure 3.4. First the learner makes its prediction from the current input values $\vec{x}$ to derive $\hat{y}$. Many machine learning algorithms require data to be normalized, however knowledge is represented by their true values. In order to determine if knowledge exists for the current inputs they must be denormalized so they can be compared to the knowledge base. If knowledge exists about the current input values the fusion of the learners output, confidence and knowledge will take place.

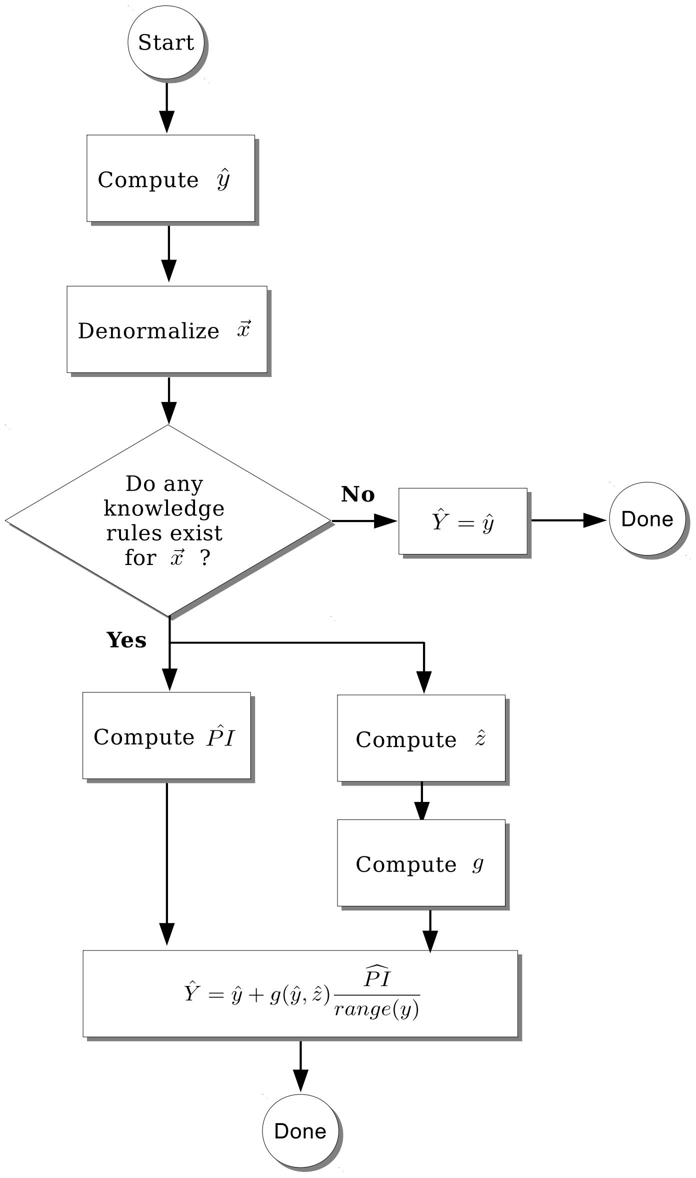

Figure 3.4: Flow chart describing how ALINK makes a prediction. In order for ALINK to utilize additional knowledge in its prediction it must first make sure that the input values in $\vec{x}$ have knowledge associated with them. If no knowledge exists then the output is simply the output $\hat{y}$ of the learner.

A question may be, if knowledge rules exist for given inputs, why even consider the estimation $\hat{y}$ of the learner so $\hat{Y} = \hat{z}$? If knowledge existed for every single input parameter, for every possible value than this may be an option however as previously discussed, ALINK's flexibility allows for select knowledge about parameters thus we must use the learner to provide additional information not covered by the knowledge. Additionally knowledge may be fuzzy and there will be a point in which the learner may be more accurate than the fuzzy system. ALINK's goal is to incorporate all available knowledge and information to make the most accurate prediction.

3.4 Experimental Implementation

ALINK implementation to be used for the extent of this thesis utilizes a fuzzy system for the knowledge base and a bagged NN as the inductive machine learning algorithm and PI method. As domain experts typically communicate there knowledge with vague linguistic terms, it is an attractive option to translate their knowledge into fuzzy logic. The added benefit of using bagged NN is it allows us to use bootstrapping as our method for deriving prediction intervals. ALINK implementation is written in Java to be run cross platform and take advantage of the numerous open source libraries available.

3.4.1 Learner

Typically a single estimator is more uncertain than an aggregation of estimators as the estimators may be constructed with non-optimal initial conditions or suffer from overfitting based on the partitioning of the data set [69, 80]. It has been demonstrated in multiple examples [10, 29, 80] that an aggregation of predictors can provide for increase in accuracy, improved generalization and robustness when compared to a single predictor. ALINK implementation uses an aggregation of NNs to take advantage of these properties. The construction of this model is displayed in Figure 3.5.

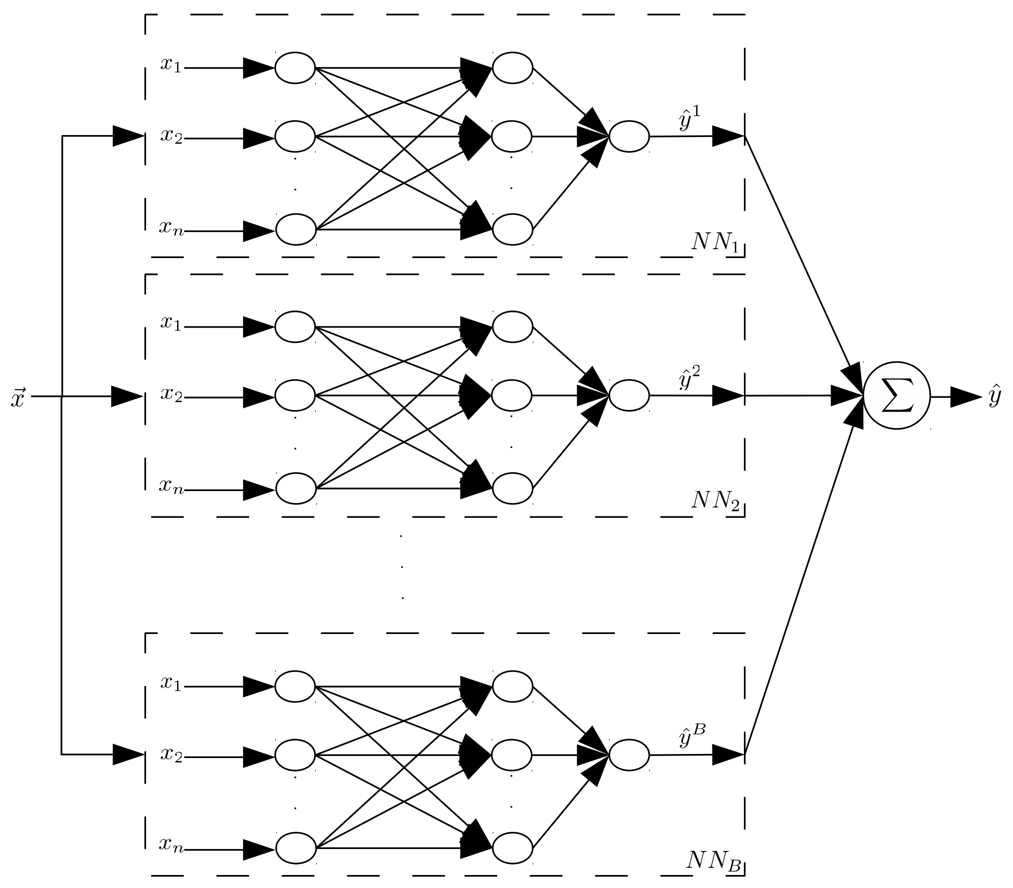

Figure 3.5: Illustration of an ensemble of neural networks. The input is fed to each network in the ensemble. The output $\hat{y}$ is the average of all network outputs in the ensemble.

Each neural network is constructed using the Java distribution of the Encog machine learning framework. Encog provides for a wide range of machine learning algorithms and data management tools.

3.4.2 Knowledge

Fuzzy logic provides a way to represent logic as an approximation rather than the strict truth. Linguistic variables are assigned linguistic terms which define a range of values rather than a typical crisp value. There must be a linguistic variable for each input and output of the system. Each linguistic variable is represented by a membership function describing all possible linguistic terms the variable can equal. IF-THEN rules are formed by pairing a linguistic variable to a term for the input and output to govern the system. For example a fuzzy logic controller for a fan may have a rule **IF** temperature **IS** hot **THEN** speed **IS** fast. Here temperature and speed are variables and hot and fast are the terms respectively.

A fuzzy system, Figure 3.6, consists of four primary components, the fuzzifier, inference mechanism, rule-base and defuzzifier [59]. During fuzzification, crisp values are converted into fuzzy values. The rule-base is a collection of "IF-THEN" statements. The inference mechanism interprets the fuzzy input and determines which knowledge to apply. Defuzzification then converts the output of the inference back into a crisp value.

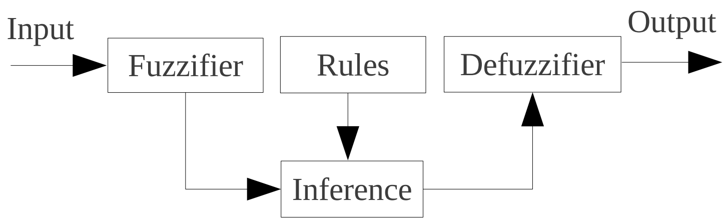

Figure 3.6: Overview of fuzzy system architechure.

Many different methods for defuzzification exist such as Centre of Gravity (COG), Centre of Area (COA), Left Most Maximum (LM) and Right Most Maximum (RM) [59]. COG is the method used in this implementation for defuzzification (3.27) where u is the output variable $\mu(u)$ is the membership function for variable u and Min and Max are the defuzzification bounds.

$$U = \frac{\int_{Min}^{Max} u\mu(u)du}{\int_{Min}^{Max} \mu(u)du} \tag{3.27}$$

ALINK uses the jFuzzyLogic [16] Java library to handle the fuzzy system. jFuzzyLogic implements the fuzzy control language (FCL) IEC-61131-7 specification which allows for easily maintaining a knowledge base defined in a FCL file. Once jFuzzylogic is integrated into ALINK, the only future modifications needed to be made are to the FCL file. This file defines all variables, membership functions and rules to be used by the fuzzy system.

Relating to Figure 3.4, before the input values $\vec{x}$ can be used by the fuzzy system they must be denormalized. Each normalized value x^* is denormalized to x according to (3.28) where d_L and d_H are the minimum and maximum values x represents in the training set, n_L and n_H are the low and high bound normalized values respectively. This is achieved using Encog's Analyst. The Analyst is a module that performs analytic tasks on data such as normalization. Before training occurs the training data is analyzed by Encog and produces a report containing a metrics of the data such as minimum and maximum values of each parameter. The fuzzy system component uses the Analyst report during predictions to denormalize the input so it can be interpreted by the fuzzy system.

$$x = \frac{(d_L - d_H)x^* - (n_H d_L) + (n_L d_H)}{n_L - n_H} \tag{3.28}$$

3.4.3 Confidence

Reference [3] claims that bootstrapping can be helpful when:

- the theoretical distribution or a statistic is complicated or unknown
- the sample sample size is insufficient for straightforward statistical inference
- power calculations have to be performed, and a small pilot sample is available

Primarily because ALINK is most beneficial for small data sets where gaps may occur in data, bootstrapping is chosen as the method to construct PIs to determine the confidence of the learner.

3.4.4 Supervisor

The degree of assistance the supervisor provides can now be defined in detail as we have defined our learner as a bagged NN and a fuzzy system as the knowledge base. The gravitational pull (3.25) introduced in Section 3.3 is now redefined as (3.29), where the slope $m = 1$.

$$g(\hat{y}, U) = U - \hat{y} \tag{3.29}$$

Further substituting of (3.26) with our selected implementations we derive (3.30) as ALINKs output. Since PIs found by bootstrap are symmetric, $\widehat{PI}$ equals the right hand side of (3.23). The data is to be normalized between [-1, 1] resulting in $range(y) = 2$ and we are looking for a 95% confidence interval in which α=0.05.

$$\hat{Y} = \hat{y} + g(\hat{y}, U)\frac{t_{0.25[v]}\sqrt{\sigma_{\hat{y}}^2 + \sigma_{\epsilon}^2}}{2} \tag{3.30}$$

Chapter 4

Experiments and Results

4.1 Defining Nerve Guidance Conduit Performance

ALINK is most beneficial in scenarios where domain knowledge exists and the sample collection is small. To test the performance benefits of ALINK under these conditions, the prediction of NGC performance is an ideal candidate as domain knowledge has been collected from conclusions drawn from the various experiments and the sample size is small as a single experiments may take as long as 60 weeks to complete [13].

Severe peripheral nerve injuries require surgical intervention to restore functionality. Currently the standard method for repair for nerve gaps > 4mm is to perform an autograft [79]. An autograft is a method in which a segment of nerve is removed from a secondary site and applied to the site of injury. However there are many risks involved in this procedure such as damage occurring to the secondary site or compatibility issues between the injured nerve and the transplant.

A NGC attempts to eliminate these risks, it is a tubular device used to bridge together the damaged peripheral nerve. Currently a variety of different construction methods are being experimented upon to try and develop an optimal construction to allow for the nerve to be properly regenerated. In order to compare the performance of different construction methods a number of different measurements have been devised including %N, L/L_c and ΔL [76].

%N is the success of regeneration of the nerve from study in which the nerve bridged the full gap. This dimensionless measure is ideal for comparison as it being a percent has known bounds which will be beneficial when normalizing data. However %N is not always reported and the way to derive %N may fluctuate between studies, reported values may be bias as different studies perform a different number of experiments.

L/L_c is the ratio of the gap length of the peripheral nerve injury, L, and the critical axon elongation L_c. L_c is obtained from a characteristic S-curve. The S-curve is found by plotting %N of an NGC versus gap lengths for various experiments. The gap length in which %N=50 is L_c. L/L_c acts as a normalization function to allow comparison between different species however it is assumed the L_c of the specie is the same which may not always be the case. Furthermore the measure L_c does not have known bounds which can pose issues when normalizing a data set.

The difference between the reported L/L_c of two NGCs is denoted ΔL. Typically ΔL is used as a measurement to determine how well the NGC performed compared to a standard conduit. A standard conduit is the simplest of conduits in which no factors are added to enhance its nerve regenerative activity. The standard conduit provides a control in the experiment. Although ΔL is useful to compare performance across multi-species and addresses the issues L/L_c has with L_c not always being the same within the specie, it suffers from the same issue as L/L_c not having an upper bound. The equation to compute ΔL is defined in (4.1) where L/L_c^{exp} is the L/L_c value of the experimental conduit and L/L_c^{std} is the L/L_c of the standard conduit.

$$\Delta L = L/L_c^{exp} - L/L_c^{std} \tag{4.1}$$

If a prediction model is accurate, it can aid in developing successful construction strategies which can potentially result in saving time, money and resources bringing the product to the market. My colleagues and I have previously used bagged NN to predict NGC performance [47], this thesis work expands on this research by incorporating collected domain knowledge.

4.2 Data Preparation

Data was extracted from 28 scientific publications in which the gap length of the injury, construction methods, performance results of the experimental conduit compared to a standard conduit and experimental run time was reported [4, 5, 9, 12–15, 18, 20, 21, 25, 27, 33, 34, 36, 38–40, 49, 51, 57, 67, 71, 73–75, 77, 79].

The run time (in weeks) in which each experiment was conducted is a crucial variable. Under proper conditions given unlimited time and resources the nerve should be able to bridge the gap. Reporting the length the nerve grew does not provide enough information to be able to make a proper prediction of the performance of the NGC. It is not until we are given perspective as to how long it actually took to grow the nerve said length that we are able to develop a forecast. Some experiments may be cut short or not allow the nerve to grow the required time to bridge the gap. For this reason time is defined as an output to the prediction model. Not only will an indication of how well the nerve grew be reported but also the expected time period these results will occur.

Referring to the previously discussed methods of determining a measure for NGC performance, ΔL is selected as the second output to the prediction model representing how well the NGC performed. ΔL allows for a comparative measure of performance across different species and poses the least number of issues to be used by a learner. The primary disadvantage of use ΔL is the unknown upper bound. This will require the data set to be re-normalized if new records are add in which the ΔL exceeds the current max value of ΔL.

138 cases were extracted from literature. Of these cases, 40 input parameters

were identified to be significant in determining the performance of a NGC. These input parameters include construction methods, materials, growth factors and other properties which affect the nerve regenerative activity. A complete list of input parameters used in the model are displayed in Table 4.1. Pre-processing was conducted to removed duplicate case, those cases in which every input parameters was identical to other cases. This resulted in 109 cases which were used for training ALINK learner and testing ALINK.

Table 4.1: The input parameters categorized based on their application to the development and enhancement of NGCs

Category	Model Input Parameters
Materials Processing	Phase Separation, Hydrogels, Electrospinning, Reverse Plating, Micropatterning, Liquid Filled
Structure	Fiber-Aligned, Fiber-Random, Gel, Permeable, Solid Tube (Impermeable), Microsphere, Porous, Internal Diameter, Wall Thickness, Lumen
Materials	Collagen, Ethyl Vinyl Acetate, Polycaprolactone, Poly Lactide, Poly Glycolic Acid, Poly Lactide Co Glycotide, Chitosan, Poly Phosphazene, Poly Pyrrole, Poly Sulfone, Silicone
Form	Hydrogel, Liquid, Gel, Matrix, Fiber-Aligned, Fiber-Random, Microsphere, Solid
Growth Factors	NGF, NT3, BDNF, CNTF, GDNF, PDGF, VEGF, FGF, Denatured FGF, IGF, Laminin, Fibronectin, Schwann Cells, Bone Marrow Stromal Cells, Neural Crest Stem Cells, Fibroblasts, α1-GP
Growth Factor Arrangements	Gradients or Anisotropic, Isotropic

Each value x was normalized to x^* between -1 and 1 using $\min - \max$ normalization according to (4.3) where d_L is the minimum value parameter x can represent, d_H is the maximum value parameter x can represent, n_L is the normalized lower bound, -1, and n_H is the normalized upper bound, 1.

$$
\begin{aligned}
x^* &= \frac{x - d_L}{d_H - d_L} \times (n_H - n_L) + n_L && (4.2) \\
&= 2\left(\frac{x - d_L}{d_H - d_L}\right) - 1 && (4.3)
\end{aligned}
$$

In order to attempt to decrease complexity of the model, categories in which all parameters contained boolean values were encoded in a binary encoding. Boolean values are those parameters indicating whether or not a parameter was used in construction of the NGC. Pruning parameters was avoided to prevent the loss of information for the explanation of performance of a conduit. All possible combinations of the boolean inputs were derived from literature review and general knowledge of peripheral nerve regeneration through application of NGCs.

The input combinations were used for condensation to keep the number of inputs to the prediction model to a minimum, thus reducing the search space resulting in a higher obtainable accuracy. The possible combinations can be seen in Table 4.2. The combinations shown are derived from general knowledge as well as the articles used to develop the dataset used to develop and support the prediction model. As advances are made in the field and complexity of combining materials advances this will be revisited.

In addition to extracting data from literature to train ALINKs learner, knowledge rules were also obtained. These knowledge rules were the result of authors making conclusions about specific construction methods they used during experimentation. The collection of knowledge rules from the various scientific publications

Table 4.2: The possible combinations for categorical fields in which parameters are boolean types. The identification of combinations allows for parameter condensing and to be represented in a binary encoding scheme.

Category	Possible Combinations
Materials	{Collagen}, {Silicon}, {Ethyl Vinyl Acetate}, {Polycaprolactone}, {Polylactide}, {Polylactide co-glycolactide},{Polysulfone}, {Poly-pyrrole}, {Polypyrrole,Polycaprolactone}, {Polypyrrole,Polylactide}, {Polycaprolactone, Polylactide}
Materials Processing	{Hydrogels},{Liquid Filled}
Form	{Matrix}, {Gel}, {Liquid}, {Matrix,Microspheres}, {Gel, Microspheres}, {Liquid,Microspheres}
Growth Factor Arrangements	{Isotropic},{Anisotropic, Gradient}

served as the knowledge base for ALINKs fuzzy system knowledge component. Each extracted rule from literature had to be converted into a fuzzy rule,

1. **IF** schwann cell count **IS** increasing **THEN** ΔL **IS** larger [5]

2. **IF** bone marrow stromal cell count **IS** increasing **THEN** ΔL **IS** med [14]

3. **IF** pore diameter **IS** small **THEN** ΔL **IS** medium high [12, 13, 57, 67]

4. **IF** pore diameter **IS** large **THEN** ΔL **IS** larger [12, 13, 57, 67]

5. **IF** pore diameter **IS** very large **THEN** ΔL **IS** low [12, 13, 57, 67]

6. **IF** c-ions **IS** low **AND** b-FGF **IS** low **THEN** ΔL **IS** larger [36]

7. **IF** BSA **IS** present **AND** b-FGF **IS** medium **THEN** ΔL **IS** larger [4, 36]

8. **IF** (NGF **IS** low) **OR** (NGF **IS** med) **OR** (NGF **IS** high **AND** NGF Delivery **IS** step **AND** laminin **IS** low) **OR** (laminin **IS** medium) **OR** (laminin **IS** high **AND** laminin application **IS** step) **OR** (laminin application **IS** continuous) **THEN** ΔL **IS** medium [21, 49]

9. **IF** (NGF **IS** low) **OR** (NGF **IS** high **AND** NGF Delivery **IS** uniform) **THEN** ΔL **IS** low [21, 73]

Variables in the knowledge rule were identified and membership functions created to define the possible known values the variable could represent. Each rule defined the expected ΔL based on the particular combinations of construction methods.

However due to the limited number of conclusions draw from literature the derived membership functions are incomplete. Few values were tested for input parameters and as a result the output given for input parameter values outside of these tested values is uncertain. Additionally since the rules are selective and do not exist for every input parameter there is risk that there may be input parameters which do not have rules causing unexpected results. The membership functions are displayed in Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, and 4.12.

This knowledge system is meant to serve as an initial proof of concept, as advances in the field continue membership functions will continue to evolve. It is also important to note that the terms assigned to membership functions may not be the true representation. For example the concentration of NGF defined to be low in Figure 4.10 may not truly be low compared to all other possible concentrations. This may be more clearly illustrated by c-ions, Figure 4.6, where the only experimented value is $30 \times 10^{14}/mm^2$. The value $30 \times 10^{14}/mm^2$ may not be low, it may actually be high for this type of application. It is not until further experimentation is done in the field that the membership functions and terms will be more clearly defined.

The accuracy ALINK reports increases as the number of records with corresponding domain knowledge increases. This of course is only valid if the rules are high quality. Of the 109 samples extracted from literature 17 of these were applicable to use the knowledge component in ALINK. As previously mentioned knowledge does not exist for all 40 parameters, ALINK will use all available knowledge to make the most accurate prediction.

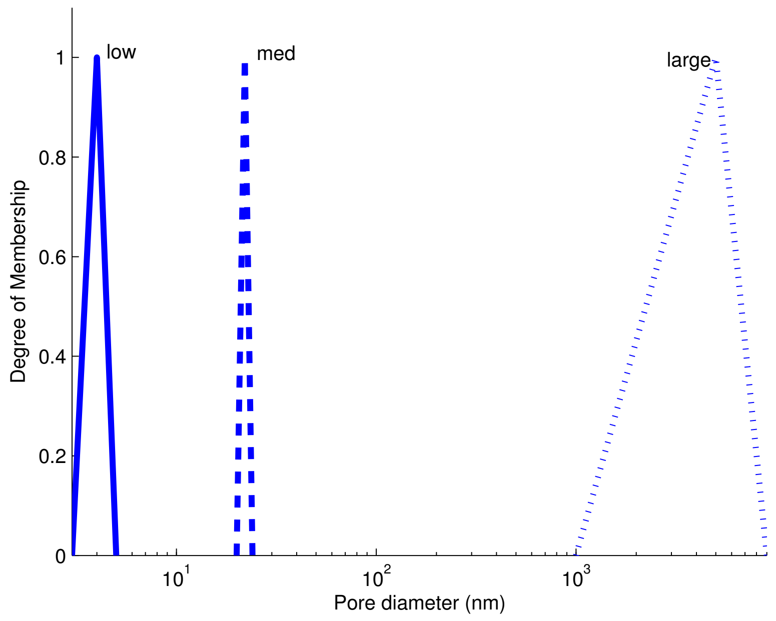

Figure 4.1: Membership function for input parameter pore diameter. Data extracted from [12, 13, 57, 67].

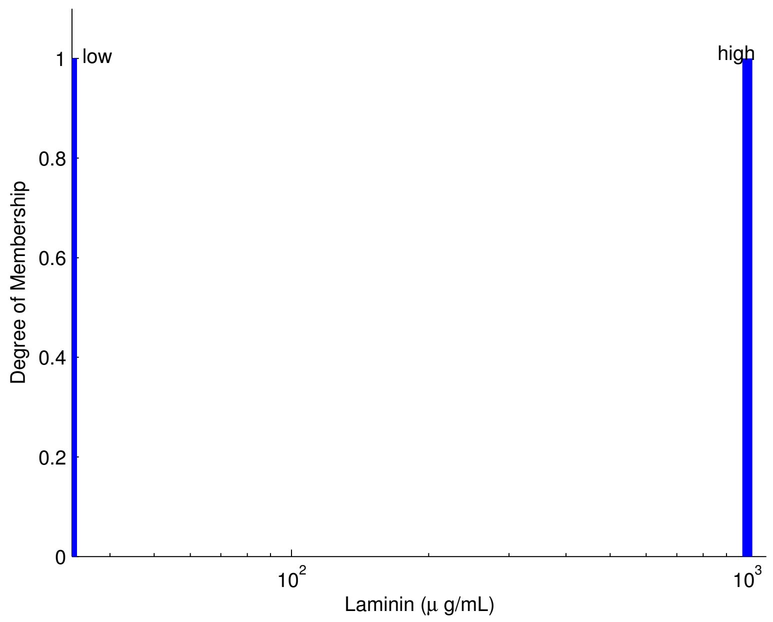

Figure 4.2: Membership function for input parameter laminin concentration. Data extracted from [21, 49].

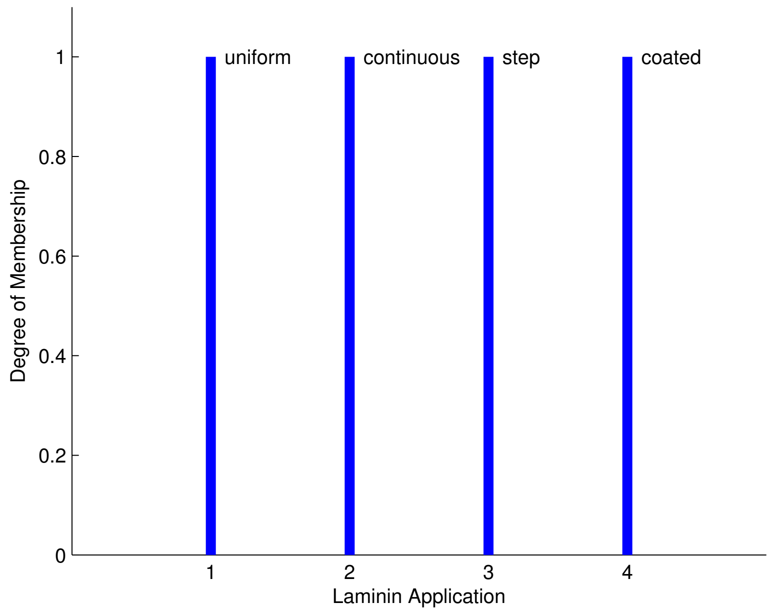

Figure 4.3: Membership function for input parameter laminin application. Data extracted from [21, 49].

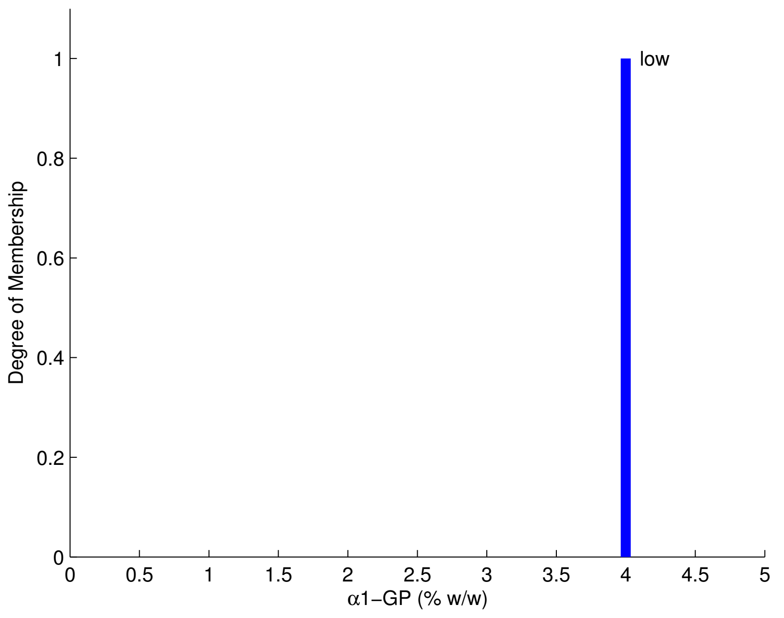

Figure 4.4: Membership function for input parameter α1-GP. Data extracted from [4].

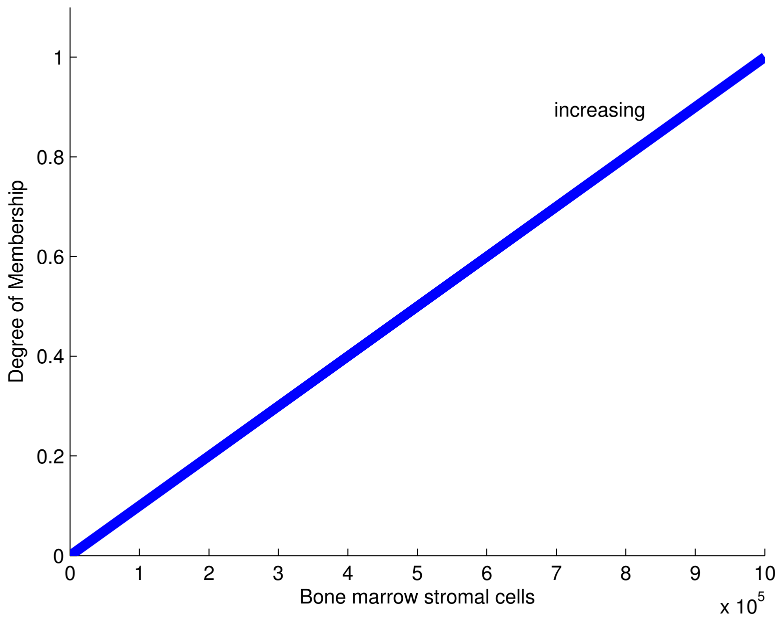

Figure 4.5: Membership function for input parameter bone marrow stromal cells. Data extracted from [14].

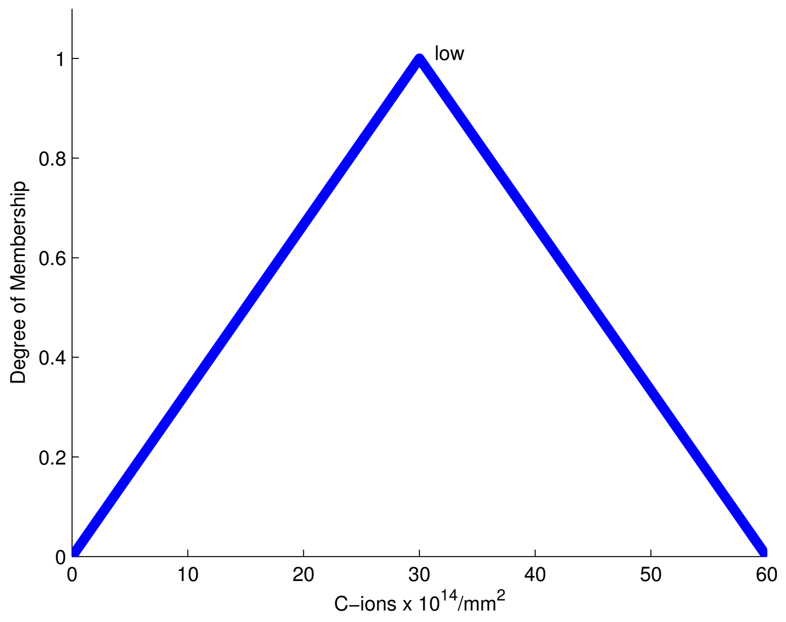

Figure 4.6: Membership function for input parameter c-ions. Data extracted from [36].

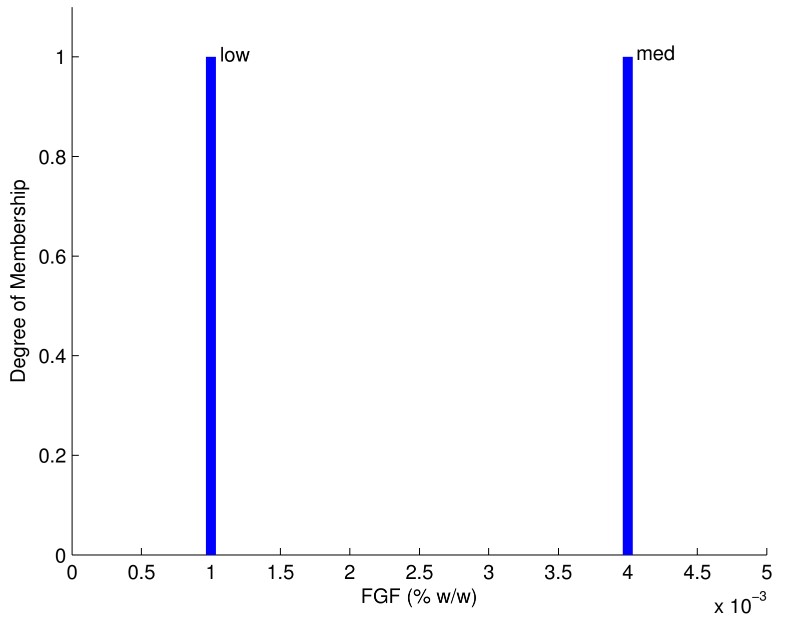

Figure 4.7: Membership function for the input parameter b-FGF. Data extracted from [36].

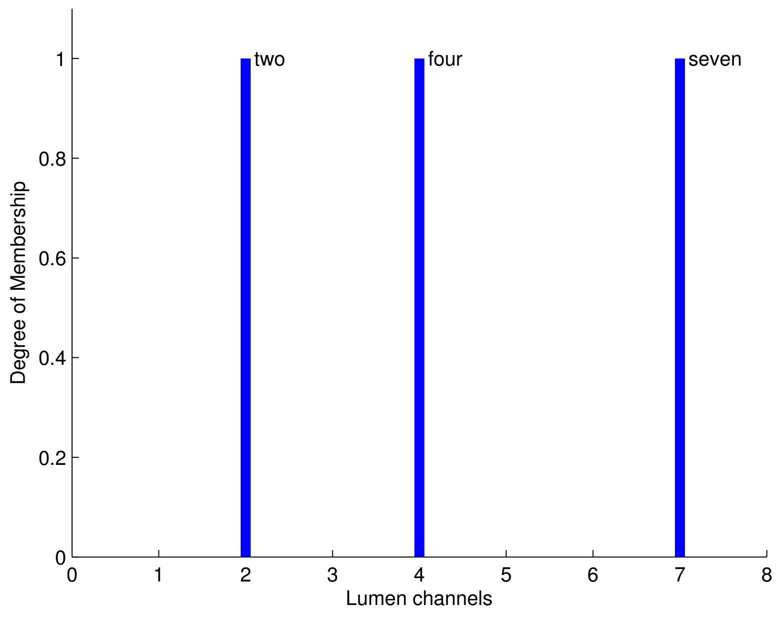

Figure 4.8: Membership function for input parameter lumen channel count. Data extracted from [13, 39, 40, 57, 67, 71, 75, 77, 79]

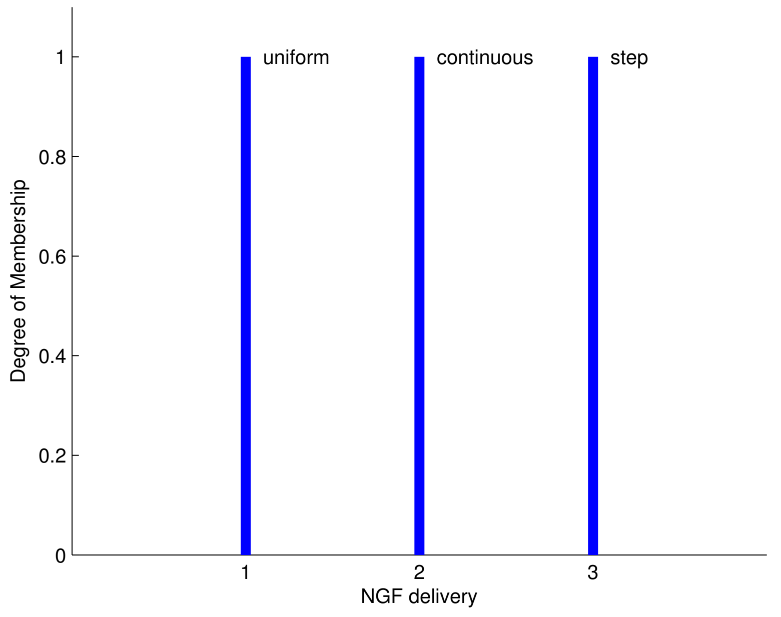

Figure 4.9: Membership function for input parameter NGF delivery. Data extracted from [21, 49, 73]

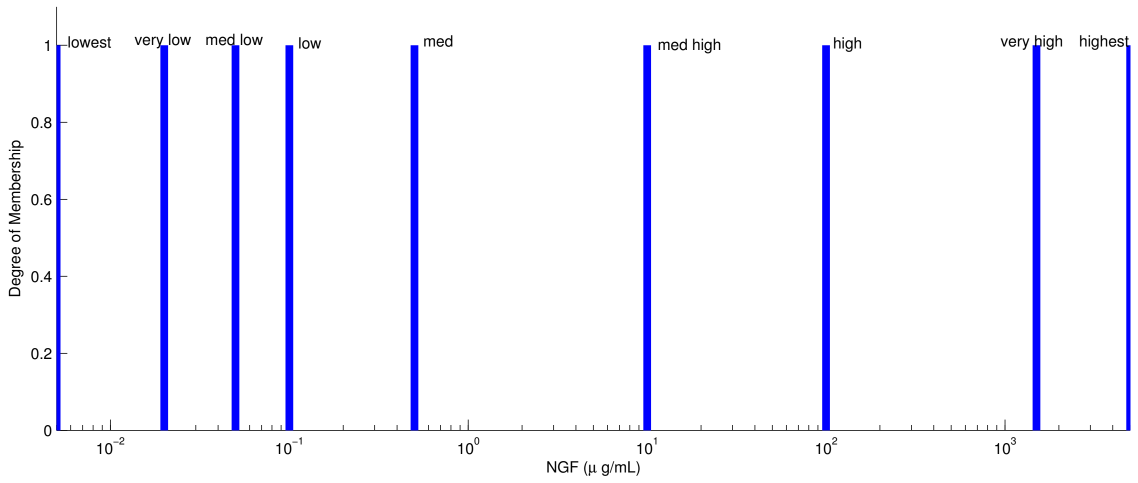

Figure 4.10: Membership function for input parameter NGF. Data extracted from [21, 49]

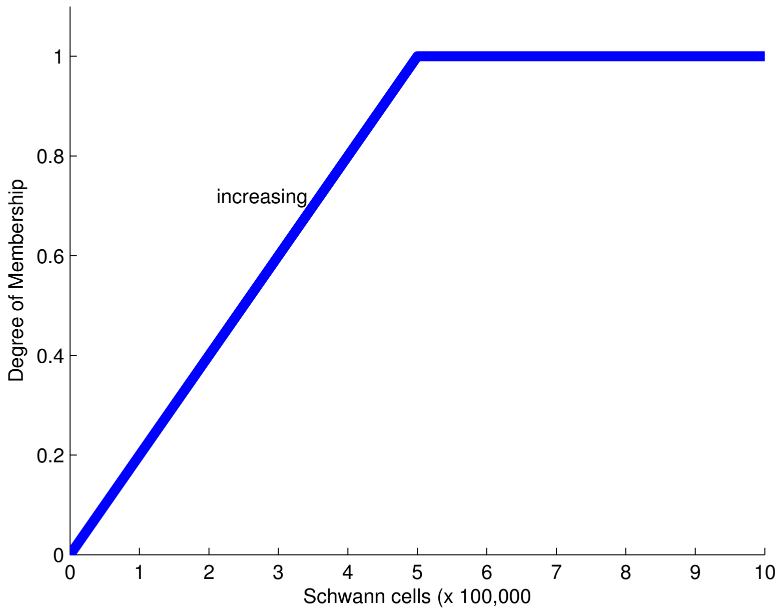

Figure 4.11: Membership function for input parameter Schwann cells. Data extracted from [5].

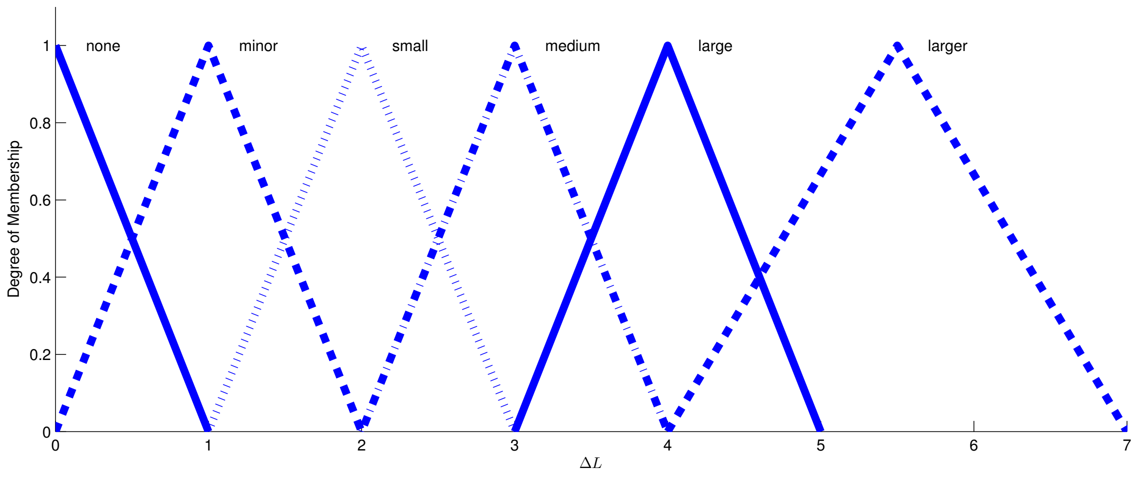

Figure 4.12: Membership function for the output ΔL. As mentioned in Section 4.1 the unbound nature of ΔL is a disadvantage of using this representation therfore membership functions are only created based on experimental evidence. The collected records were clustered to determine the number of terms needed.

4.3 Experimental Setup

To test the accuracy of the ALINK predicting NGC performance, we compare it to the accuracy of a FFNN, a bagged NN, and SWarm Intelligence based Reinforcement Learning (SWIRL).

SWIRL [17], which seems to be the only other model attempting to predict NGC performance, is used as comparison. SWIRL is a school system model where ant colony optimization (ACO) [22] acts as the administrator, particle swarm optimization (PSO) [44] represents the teachers and each NN in an ensemble acts as a student. ACO essentially models the navigation behaviour of ants and is applied in this context to find the optimal NN topology and PSO is used to train each of the NN students. PSO is a population based optimization algorithm in which particles evolve in search-space to find solutions to a problem. Once SWIRL finds the optimal NN it can be used for testing. The SWIRL model was configured with 100 students, PSO parameters of cognitive and social learning rates set to 2.0 and inertia weight set to 0.8, and ACO parameters of pheromone influence factor $\alpha = 2.0$, desirability influence factor $\beta = 1.0$, and pheromone persistence $\rho = 0.5$. The SWIRL model uses one hidden layer where hidden units range from 3 to 15. Linear transfer functions were used for input units and hyperbolic tangent transfer functions were used for hidden and output units.

The remainder of the models are trained with BP and PSO. Backpropagation is chosen as a benchmark since it is the most commonly used training algorithm. The learning rate was set to 0.01 and the momentum to 0.95. PSO was configured to have an initial population of 40, cognitive and social learning rates set to 2.0 and inertia weight set to 0.4. Max velocity was limited to 2.0.

Tests were conducted to find the optimal network topology for each training strategy with the exception of SWIRL since its topology is dynamic. According to [30], one hidden layer should be sufficient for modelling most scenarios however two may be needed for more complex situations. One of many rules of thumb for determining hidden node count is to have the count between the number of input nodes and number of output nodes. Based on this information network topologies of one and two hidden layers are investigated.

For networks trained with backpropagation, the hidden nodes for each layer range between 1 and 40 nodes and when trained with PSO, the hidden nodes for each layer range between 1 and 10 nodes. The number of hidden nodes is limited to 10 when trained with PSO based on previous experience of not seeing added benefit above this number that come with additional computational expense. 50 instances of the data set are randomly partitioned at the beginning of the experiment so each topology is trained and tested on each of the 50 instances to ensure results reflect topology performance. The average mean squared error (MSE), μ_{MSE}, of the 50 trials is determined for each topology.

Results displayed in Fig. 4.14 conclude that the optimal topology for a network trained with backpropagation for this data set is a network with two hidden layers, the first with 20 hidden nodes and the second with 1 hidden node. The results for training with PSO displayed in Figure 4.15 conclude a single hidden layer with 5 nodes is optimal. From this point forward experimentation using networks trained with backpropagation are implemented with a network topology of 40-20-1-2 and when trained with PSO are implemented with a topology of 40-5-2.

The network input units were constructed with linear transfer functions (4.4) while hyperbolic tangent transfer functions (4.5) were used for the hidden and output units. Bias units are used for hidden and output layers.

$$\phi(x) = x \tag{4.4}$$

$$\phi(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4.5}$$

The NNs implemented for all models with the exception of SWIRL is displayed in Figure 4.13. However when trained with backpropagation the network has an additional hidden layer not shown.

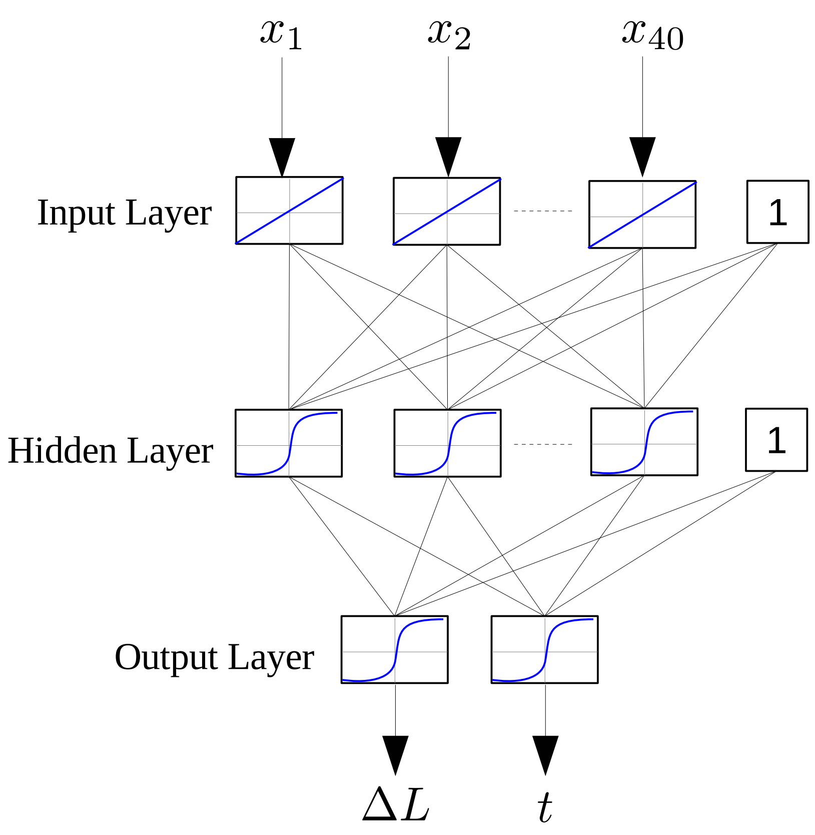

Figure 4.13: Neural network implementation used for predicting NGC performance. Input layer consists of 40 units each constructed with linear transfer functions. Hidden and output nodes are constructed with hyperbolic tangent transfer functions. Bias units, those units labelled **1**, are used for hidden and output layers. Network outputs are ΔL and time t which ΔL can be expected.

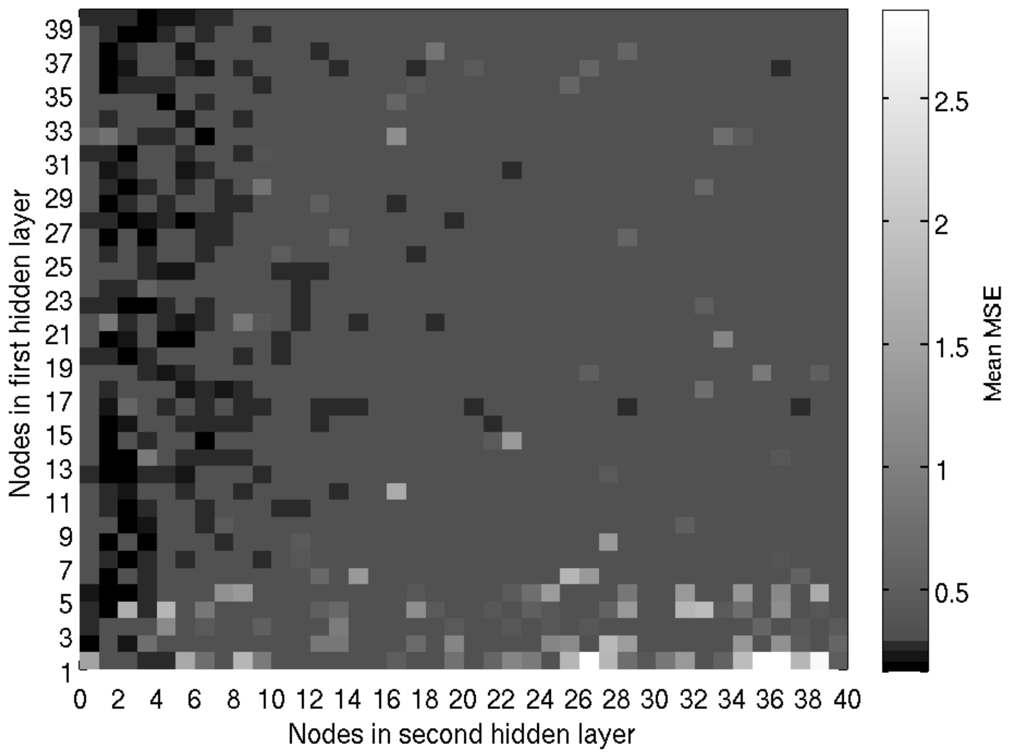

Figure 4.14: Color map displaying network topology performances when trained with backpropagation. The color bar indicates the average MSE, μ_{MSE}, of 50 trials conducted for each topology. 50 instances of the data set are randomly partitioned at the beginning of the experiment so each topology is trained and tested on each of the 50 instances to ensure results reflect topology performance. Darker represents lower error. The lowest error is acheived with 20 nodes in the first hidden layer and one node in the second hidden layer.

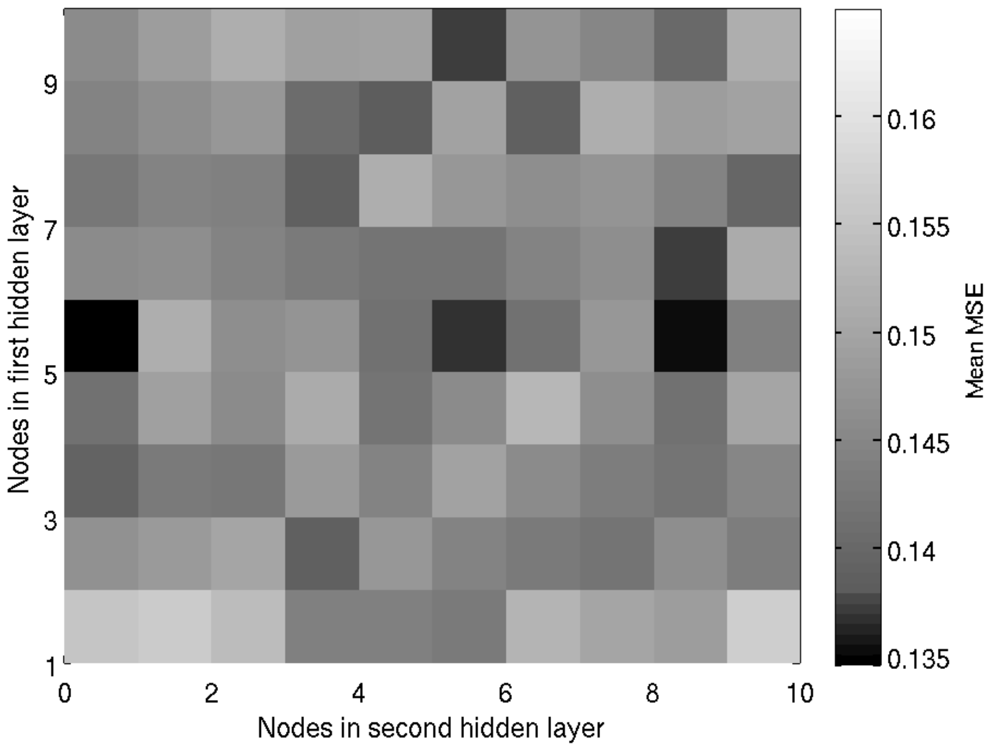

Figure 4.15: Color map displaying network topology performances when trained with PSO. The color bar indicates the average MSE, μ_{MSE}, of the 50 trials conducted for each topology. 50 instances of the data set are randomly partitioned at the beginning of the experiment so each topology is trained and tested on each of the 50 instances to ensure results reflect topology performance. Darker represents lower error. The lowest error is acheived with 5 nodes in the first hidden layer and zero nodes in the second hidden layer (i.e., the second layer is not used).

With data prepared and models configured, experimentations are ready to run. Before continuing, the bootstrap confidence implementation is tested to ensure proper functionality using a synthetic data set in Section 4.4.1. After the implementation is verified the optimal number of networks to be used in the bagged NNs needs to be determined in Section 4.4.2. The number of networks B ranged from 20 to 200 incremented by 20 which has been suggested by [23] to demonstrate the best performance. Bootstrap pairs approach is selected for sampling because we are concerned with an unconditional distribution and do not want to be limited to the collected training observations. The execution times training different ensemble size is reported in Section 4.4.3. With the optimal ensemble size determine benchmarking can be performed, results are reported in Section 4.4.5. All experimentation was executed on a 64-bit Intel® Core™i7 @ 2.20GHz processor with 8GB of RAM.

4.4 Results

4.4.1 Bagged Neural Network Prediction Intervals

To ensure proper implementation of finding PIs using the bootstrap method a synthetic experimentation was conducted similar to that found in [31]. Input values x was sampled 1000 times between [-1,1] from a U-quadratic distribution. Two parameters must be defined for this distribution, a and b, the lower and upper limits which in this experiment equal -1 and 1 respectively. The limit parameters further define β, the gravitational balance center (4.6) and α the vertical scale (4.7).

$$\beta = \frac{b+a}{2} \tag{4.6}$$

$$\alpha = \frac{12}{(b-a)^3} \tag{4.7}$$

The U-quadratic distribution has a probability distribution function (PDF) defined by (4.8) and a cumulative distribution function (CDF) defined by (4.9). Sampling from the U-quadratic distribution was achieved by extending the AbstractRealDistribution class available from the Apache Commons Mathematics Library.

$$PDF = \alpha(x-\beta)^2 \tag{4.8}$$

$$CDF = \frac{\alpha}{3}\left((x-\beta)^3 + (\beta - a)^3\right) \tag{4.9}$$

The synthetic function used to generate the target is defined by (4.10). The collected data was then randomly partitioned 5% training, 95% testing or 500 training examples and 950 examples for testing.

$$t = sin(\pi x)cos(5\pi/4)\sqrt{0.005 + 0.005(1 + sin(\pi x))^2} \tag{4.10}$$

A bagged NN was constructed with 10 networks each of which had 8 hidden nodes implemented with hyperbolic tangent activation functions while input and output nodes were implemented with linear activation functions. The neural network predicting noise was constructed with only a single hidden node.

The U-quadratic distribution was selected to clearly demonstrate the performance of deriving PIs using the bootstrap method. This sampling allows for a higher probability of samples to be drawn on the edge of the intervals while the least number of samples drawn around 0. As a result we should see a narrower PI in areas in which the model is trained with more examples and a wider PI in areas which are trained more sparsely.

Figure 4.16 is the result of finding PIs using the bootstrap method. As expected we can see that the PI is wider for values centered around $x = 0$ in which training was more sparse in these areas. This functional example provided confirmation of the proper Java implementation of the bagged NN with PI implementation providing the base to build in the additional ALINK logic.

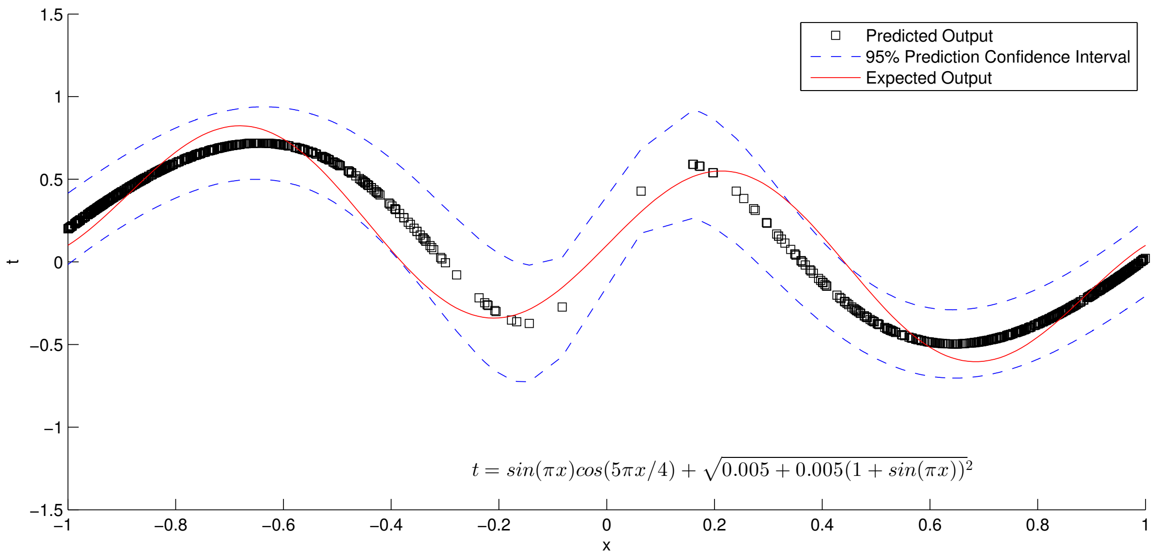

Figure 4.16: An synthetic example of defining the 95% prediction interval using the bootstrap method. A bagged NN is trained on a random U-quadratic sampling between the interval [-1,1]. We can see that areas that are more dense during training have narrower PIs indicating the model is more confident to predict these values. Sparse samples centered around 0 have wider PIs indicating the model is less confident in its prediction.

4.4.2 Optimal Ensemble Size

Before benchmarking is performed, the data set is randomly partitioned into 80% training and 20% testing. This is repeated to form 20 unique training\testing pairs. Each model in a benchmark trial is trained and tested on each pair. The MSE is produced from each test set. The average MSE from the 20 sets is then reported for comparison.

The optimal ensemble size when trained with backpropagation was found to be 100 for the NGC data set. The average MSE for each ensemble size is displayed in Figure 4.17. When the networks in the ensemble were trained with PSO the optimal number of networks found was also 100 displayed in Figure 4.18.

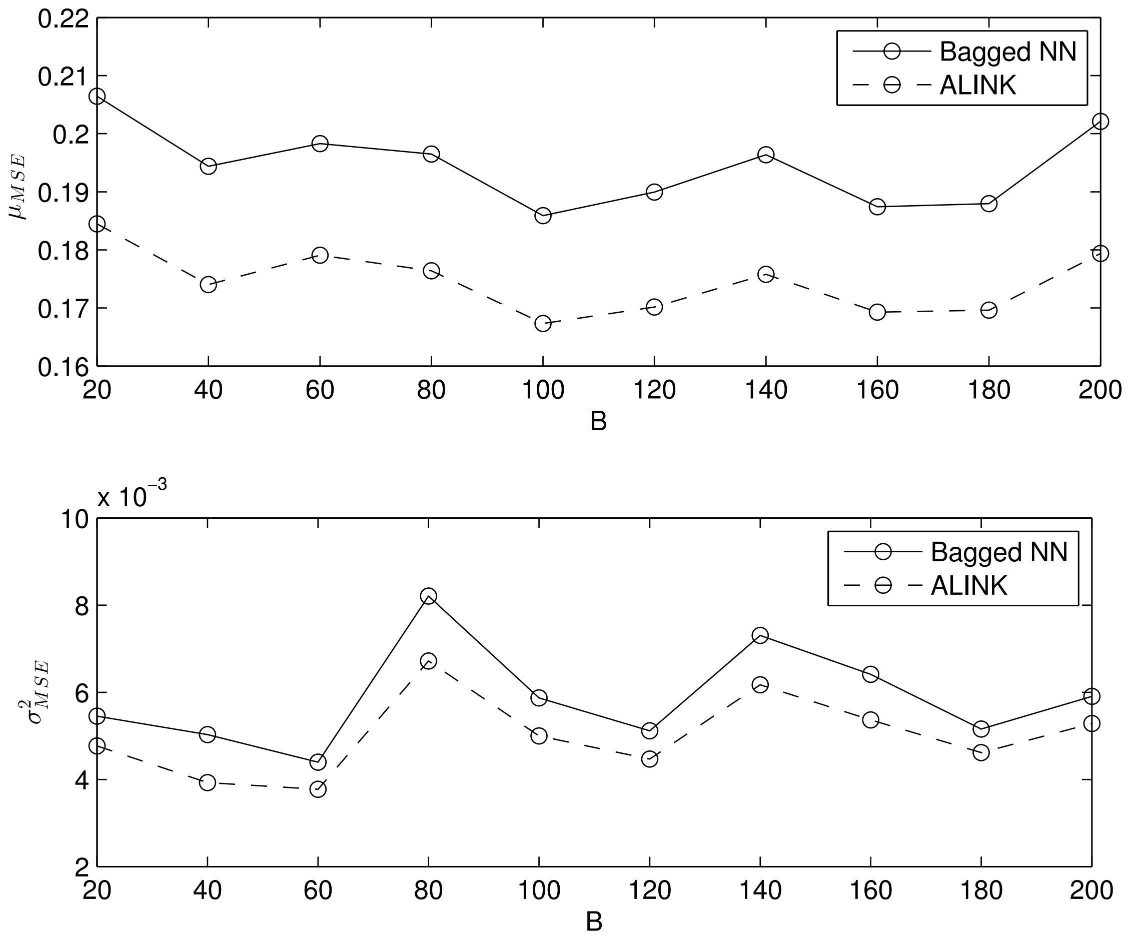

Figure 4.17: Results determining optimal number of networks to use in ensemble for bagged NN and ALINK when trained with backpropagation. Top subplot is mean MSE of 20 trials, while the variance between trials is shown in bottom plot.

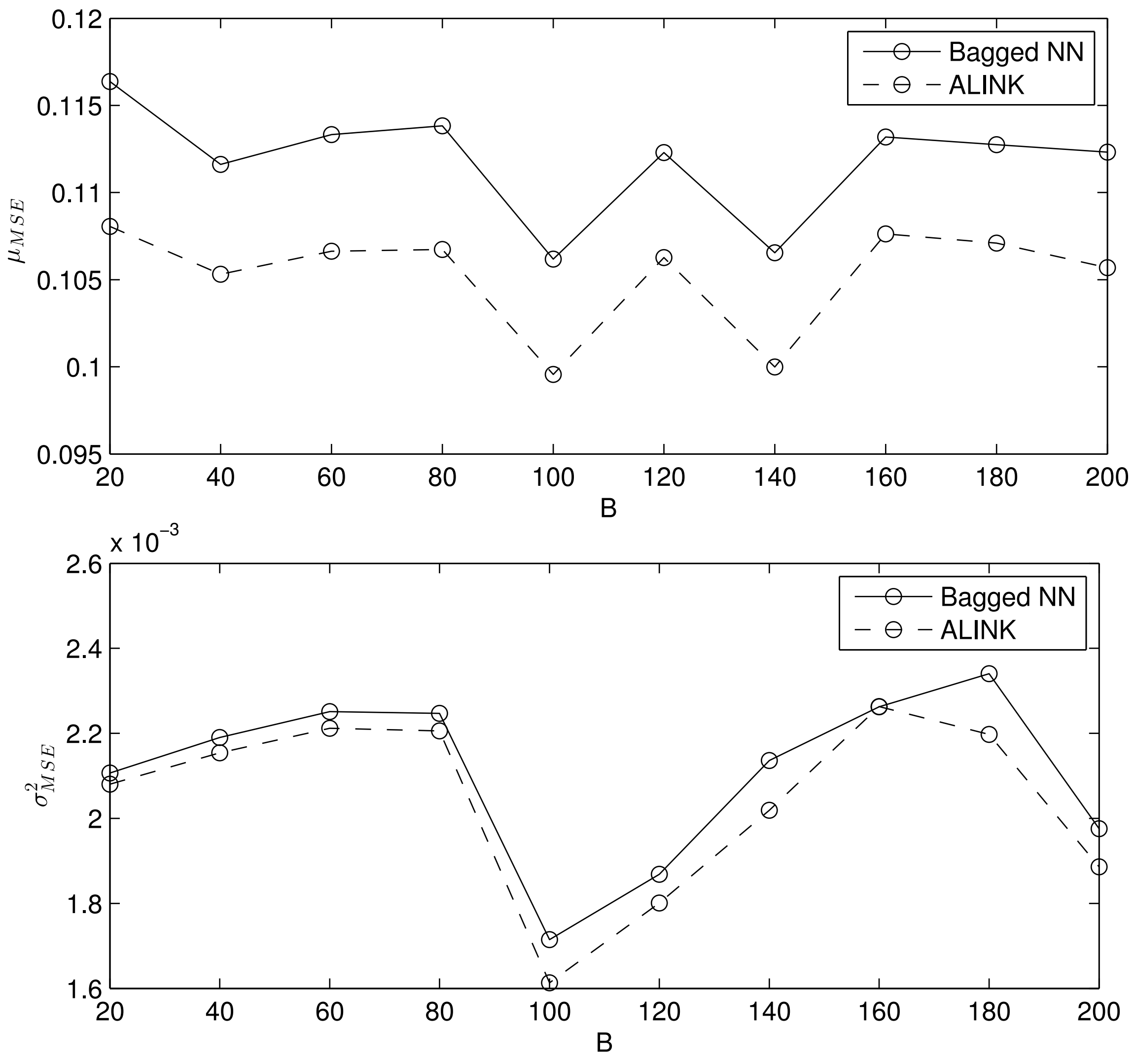

Figure 4.18: Results determining optimal number of networks to use in ensemble for bagged NN and ALINK when trained with PSO. Top subplot is mean MSE of 20 trials, while the variance between trials is shown in bottom plot.

4.4.3 Execution Runtime

The training time of the incremental ensemble sizes was recorded during execution of the experiment. In Figure 4.19 we can see that as the number of networks are added there is a linear relationship in the training time. Networks trained with PSO take about 20 times longer to train. Due to the incorporation of knowledge being independent of training there is no additional computational expense.

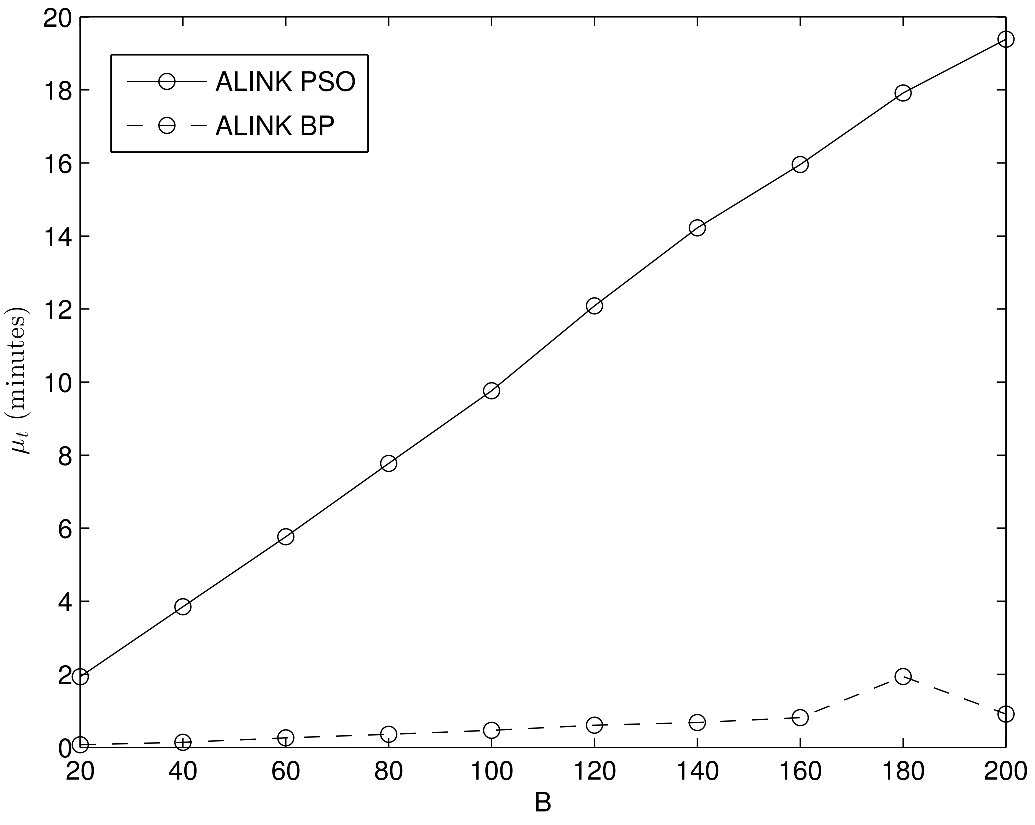

Figure 4.19: Comparison of execution times to train ALINK with backpropagation and PSO when implemented with a bagged NN. The time in minutes is an average over 20 runs.

4.4.4 ALINK Candidate Cases

As outlined in Section 4.2, only 17 of the observed cases qualify as candidates to test ALINK in its entirety. To illustrate how ALINK arrives at its prediction 7-fold cross validation was performed on the data set so each observation would be used during training and testing. The candidate cases were then parsed from the results which are displayed in Figure 4.20. It is interesting to note that some cases which

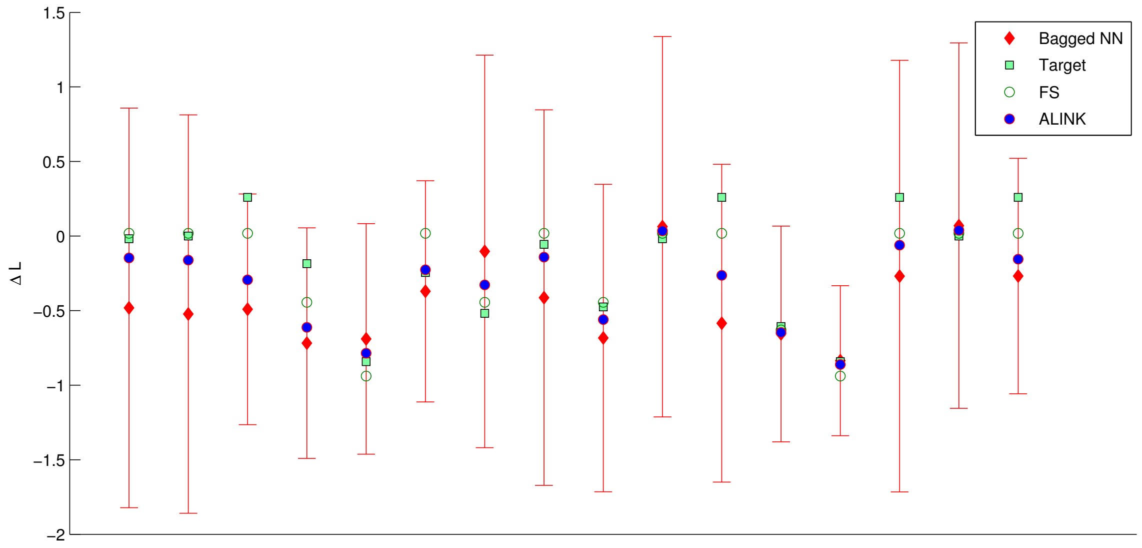

Figure 4.20: Details of ALINK deriving the output for candidate cases. The plot shows the output of the each components in the framework. The circle outline represent the output from the FS, diamonds represent the output from the bagged NN while the error bars represent the 95% PI of the bagged NN prediction. The bagged NN was constructed using the optimal size ensemble $B = 100$ trained with PSO. Finally the output from ALINK is represented by the solid circle. To give perspective the target ΔL output is represented by a solid square. For cases that have wider intervals the learner output gravitates more toward the fuzzy system. This concept illustrates the gravitational pull toward the knowledge base to provide for more accurate prediction when confidence is low.

have accurate bagged NN outputs form very large PIs. This is a good indicator that other PI construction methods should be investigated as the ALINK model is only has accurate as its individual sub-components. If the PIs are inaccurate causing particularly wide PIs while the predictor is accurate, it can cause the output to be overshot gravitating strongly to the FS when in reality the learners prediction was more accurate.

4.4.5 Benchmarking

Using the optimal ensemble size, ALINK performance is compared to a FFNN, bagged NN and SWIRL in which the FFNN and bagged NN are both trained with backpropagation and PSO. Each model tested has its MSE (4.11) calculated for each of the 20 trials,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4.11}$$

where n is the number of testing examples, $\hat{y}_i$ is the predicted value by the model and y_i is the ideal value. The MSE values of each of the 20 trials is averaged,

$$\mu_{MSE} = \frac{1}{N}\sum_{i=1}^{N} MSE_i \tag{4.12}$$

This measure it was is used for comparison. The sample variance is calculated between all trials,

$$\sigma^2_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(MSE_i - \mu_{MSE})^2 \tag{4.13}$$

The results displayed in Figure 4.21 show that ALINK when trained with PSO out performs all other methods. The overall mean and variance of the MSE found from each experimental trial is displayed in detail in Table 4.3. The primary difference between the FFNN trained with PSO and SWIRL is the dynamic topology SWIRLs implements. The FFNN trained with PSO proves more accurate thus suggesting that the stochastic nature of ACO misleads SWIRL to choosing a non-optimal topology causing poor performance.

ALINK proved to show a 6% decrease in error from a standard bagged NN. Because the incorporation of additional knowledge is the distinguishing factor between these two methods this decrease in error is a reflection of the collected domain knowledge. As more knowledge is discovered about NGC performance ALINK will continue to be more accurate than a bagged NN. ALINK showed a 44% decrease in error over SWIRL demonstrating that it is able to aid in developing more accurate strategies for NGC construction than SWIRL was able to accomplish.

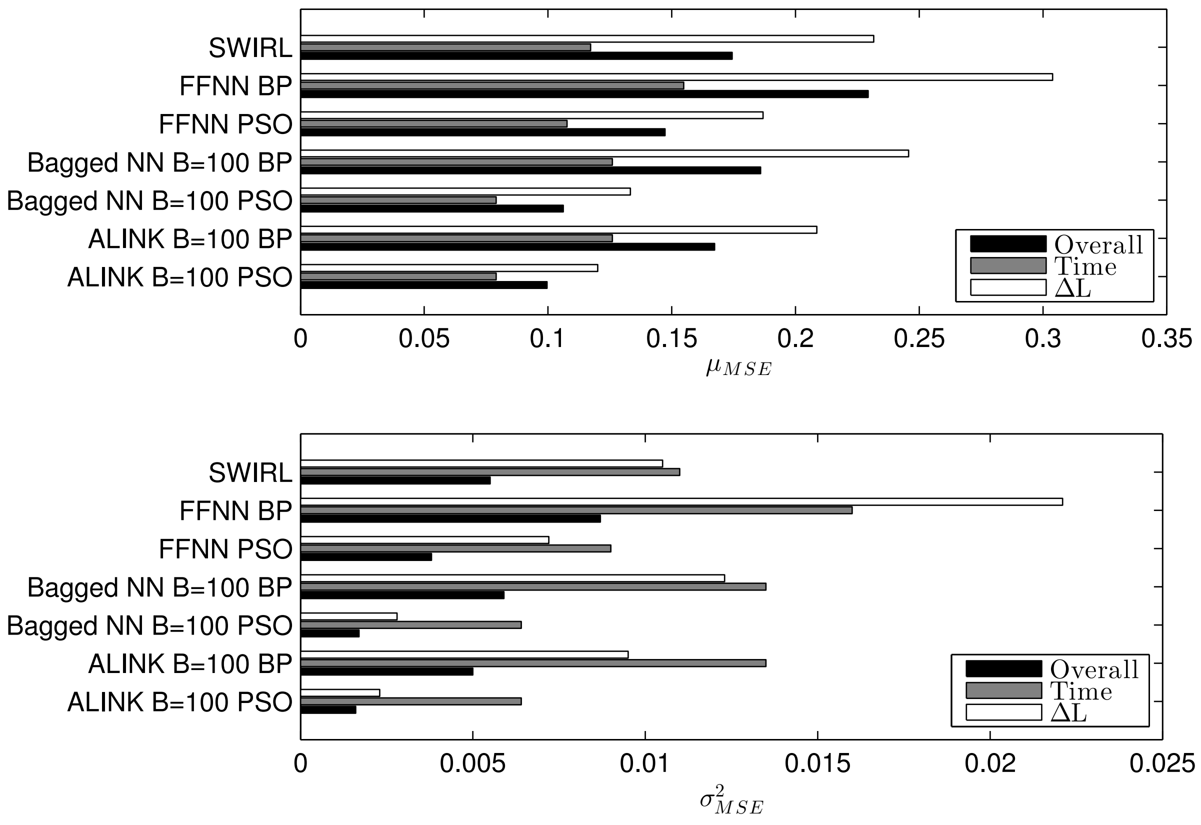

Figure 4.21: Result of models benchmarked. Top subplot is mean MSE μ_{MSE} of 20 trials performed. The variance between trials σ^2_{MSE} is shown in bottom plot.

Table 4.3: Summary of Benchmark Results

Model	μ_{MSE}	σ^2_{MSE}
SWIRL	0.1744	0.0055
FFNN, BP	0.2294	0.0087
FFNN, PSO	0.1473	0.0038
bagged NN, B=100, BP	0.1859	0.0059
bagged NN, B=100, PSO	0.1062	0.0017
ALINK, B=100, BP	0.1673	0.0050
ALINK, B=100, PSO	0.0996	0.0016

Chapter 5
Conclusion

Collecting an adequate number of samples to make accurate predictions will continue be challenging in many domains. Luckily when domain knowledge exists there are additional options that can be used to incorporate this information to increase prediction accuracy.

Virtual examples are a useful option when synthesizing new training examples are practical as the virtual examples are transparent to the learner allowing for no modification of the machine learning algorithms. This allows the developer to use 3rd party machine learning software packages. Determining the optimal way to generate synthetic training examples is the main challenge as examples with high correlation will be of little help to the learner.

Hints are an attractive option to incorporate prior domain knowledge. There are many different types of hints to choose from and the effect each has during training can be controlled independently. Hints however are integrated directly into the training algorithm. Modification of decent algorithms has been proposed however because of there experimental nature it is not common practise and as such are not included in machine learning toolboxes. To utilize hints it is left to the developer make the proper modifications.

ALINK has been developed to be beneficial in scenarios where prior or post domain knowledge exists and data is limited. Its primary objective was to serve as a framework allowing the developer to select their own learner, trainer, and knowledge base depending on the target application. It has been designed so the learners training algorithm is not affected so any training algorithm can be applied. This allows the developer to use 3rd party software out the box without modification to training algorithms. ALINKs method of fusing information for different components based on confidence is the novel method presented in this thesis. The method of measuring the confidence of a predictor gives us an idea of how well we can trust the prediction. If the predictor is not confident it should be assisted with any additional knowledge available. Furthermore by keeping knowledge integration independent of training it allows for post knowledge incorporation. When new data becomes available in the form of knowledge, rules can be modified, added or removed without the need of re-training a learner. This can be particularly useful when working with large data sets where training can be very time consuming or for devices which ALINK is deployed with minimum hardware where re-training would be unrealistic.

ALINK however is only as accurate as its subcomponents. If a poor method for confidence is selected the impact the knowledge has on the output could actually cause more harm than good. The same applies to the rules defined in the knowledge base.

ALINK has been demonstrated to be useful for modelling systems where data is collected from long-term experiments. Long-term experiments are seen in almost every domain such as tissue engineering. Predicting the performance of a NGC is a perfect example as single experiments may take as long as 60 weeks to complete [13]. This in conjunction with large number of model parameters causes the predictors to suffer from Hughs Effect [58] in which the prediction accuracy decreases as the dimensionality of the problem increases. The high dimensionality also causes the learner to be subjected to overfitting when data set size is small. Overfitting was addressed by using a bootstrap aggregated neural network learner implementation in ALINK providing for enhanced generalization capabilities. It has been demonstrated that ALINK is able to increase prediction accuracy of NGC performance by incorporating domain knowledge. Providing the additional domain knowledge allowed for the model to fill the void where training examples were sparse. As more rules are discovered about NGC construction, accuracy will continue to increase. The model outperformed a single FFNN, bagged NN and SWIRL when trained with PSO. ALINK demonstrated a 44% decrease is MSE compared to SWIRL, the only other model to be known of predicting NGC performance. Although training the bagged NN with PSO was computationally expensive the significant decrease in error out weighed the negative effects. This advancement in prediction will further enhance the progress towards developing an optimal conduit to restore function from a peripheral nerve injury.

Chapter 6
Future Work

In Section 4.1 possible output representations of NGC performance was discussed. Although ΔL was selected to be the best option, the unbound value will create problems in the current implementation if values are used outside of the collected training data. Due to the data normalized between [-1,1] and hyperbolic tangent used as the activation function for the output nodes, the predicted ΔL is limited to the max value in the training data. To address this issue alternative activation functions for output nodes will be investigated to improve robustness. For example exponential activation function have been used to enforce positive values in [45] and [53].

At this time, knowledge rules only exist for the ΔL output. To provide for a more accurate indication of when the expect ΔL will occur, knowledge rules also need to be developed for the *time* output. We already have some knowledge about the time output, given unlimited time and resources, nerve regeneration should continue. Thus given a short time we expect a small change in ΔL. Time and ΔL are strongly correlated but the exact relationship is dependent on construction parameters. This information must be correctly interpreted so it can be translated into rules to be used by ALINK.

The prediction of NGC performance is still in its infancy. Continued improvements need to be made to the knowledge rules primarily in respect to expanding the membership functions. As advances are made in the field and additional experiments are conducted, the knowledge base will continue to evolve. Additionally further research needs to be completed in regards to each input parameter. This will aid in defining membership functions more accurately and creating more appropriate terms associated with each membership function. For example it would be very beneficial to obtain the saturation point of each mixture input parameters such as laminin and NGF to be able to define bound limits.

It is imperative the collected data used in this thesis be made available to the public to continue the progression of modelling performance of NGCs. An enormous effort which spanned many months was made by my colleagues to identify publications involved in performing experiments on NGC construction, extracting available information relevant to predicting NGC performance, formatting data and compiling data into a central repository. A web application is needed to provide a portal to allow others to add data in the form of experimental results and/or knowledge rules. The central public repository will also allow for other models to be tested with this data equating to greater attention in the community to further progress construction of NGCs.

As a proof of concept, ALINKs gravitational pull was defined as linearly in (3.25). As the knowledge is intended to fill the gaps in training, if the disagreement between the output of the learner and knowledge is great it is probably an indication the estimate was inaccurate thus the output shall gravitate more favourably toward the provided knowledge. For this reason it may be more accurate to define the gravitational attraction in a non-linear fashion. In Figure 6.1 a linear, 2-degree polynomial and exponential gravitational pull are compared.

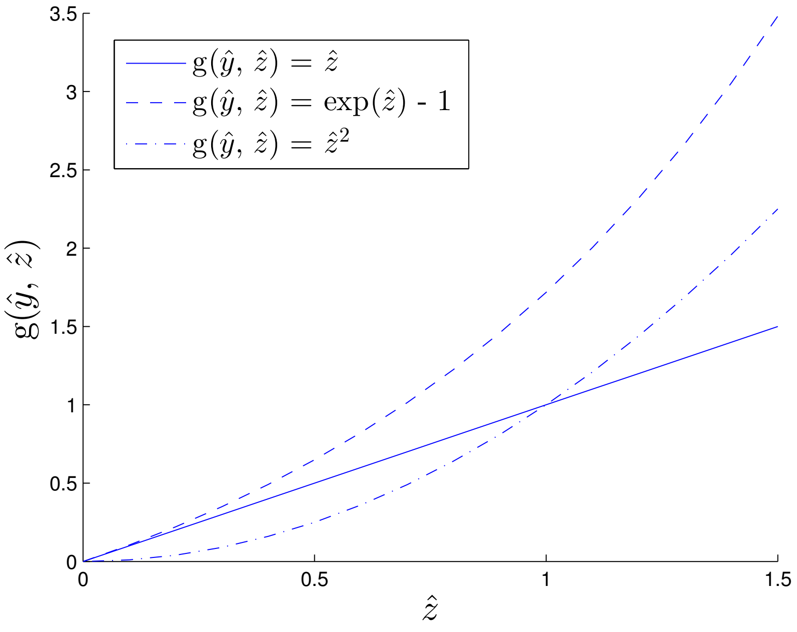

Figure 6.1: Possible examples of alternative functions for defining the gravitational pull.

Assuming in this example the estimate of the learner is $\hat{y} = 0$ and the knowledge output is $\hat{z}$ we can see the affects the different functions will have on the gravitational attraction. The linear attraction provides for a proportional increase in pull the farther $\hat{z}$ is from $\hat{y}$ however the exponential pull would provides a more aggressive approach, dramatically pulling the output to that determined by the knowledge. The effects of a 2-degree polynomial would allow the pull to remain weak until the critical point is reach in which the pull becomes increasingly stronger. Future work will investigate potential use cases for selecting different equations for defining the gravitational attraction between $\hat{z}$ and $\hat{y}$.

The focus of this thesis was to introduce a new method to more accurately estimate NGC performance. This was achieved by incorporating knowledge into a prediction based on the learners confidence. Further investigation needs to be conducted to compare the benefits of using ALINK over virtual examples and hints. Future work will include benchmarking ALINK against these other two methods. Additionally, to exam the robustness of ALINK other data sets need to be tested.

Since ALINK implementation used in this thesis uses a fuzzy system and provides a fuzzy estimate of the prediction, once the learners confidence is high, a point will eventually be reached in which the learner is more accurate than the fuzzy system. Investigation still needs to be explored as to what point the learner over takes the knowledge system and the negative affects that may result from the output always gravitating toward the knowledge system.

A primary advantage of ALINK is the ability to incorporate post domain knowledge without the need to retrain a learner which can be a time consuming task with big data. However when new observations are obtained we want to be able to teach the learner this new information. Future work will investigate the performance of implementing ALINK using online learning [43, 52, 72]. When the learner is young, trained on limited data, it will be able to use the additionally provided knowledge as a crutch for gaps in its training. As the learner grows, taught on new observations the confidence of the learner should increase relying less on the supervisor.

Assuming the reported confidence of a learner is accurate, it provides powerful insight on how the learners prediction can be used. When combining an ensemble of experts it is usually performed by selecting the best classifier or performing an average [6]. It was shown by [60] that taking the average of the highest confident networks in an ensemble 80.2% accuracy predicting secondary protein structures was achievable. By creating subsets of training data [37, 42] used divide-and-conquer to form a hierarchy of experts. A gating network is then applied to determine which expert handles which case. [32] reviews several methods for combining classifiers, among these is Dynamic Classifier Selection (DCS) in which results are achieved dynamically by a weighted average [6]. It will be investigated if an ensemble of learners can formulate a more accurate result than reported by methods in [6] by exploiting confidence of each learner to gravitate to a agreed upon output as demonstrated in ALINK.

Obtaining a sufficient number of observations to train learners will continue to be a problem in many domains. It is imperative to continue investigating methods to incorporate as much available information about a system as possible to produce the most accuracy predictions. ALINK has presented a novel method for incorporating domain knowledge based on a learners confidence, continued efforts will be focused investigating methods for exploiting confidence and domain knowledge to improve prediction accuracy in domains where data is sparse and unbalanced.

Appendices

A Acronyms

ACO ant colony optimization 45, 46

ALINK Assisting Learners by INcorporating Knowledge xi, xiv, xv, 2, 3, 13, 22, 23, 25–27, 30–32, 35, 36, 38, 39, 45, 52, 54–59

bagged bootstrap aggregated xiv, xv, 21, 27, 31, 45, 50, 52–59

BP backpropagation 46

FCL fuzzy control language 30

FFNN feed-forward neural network 9, 45, 56–58

MSE mean squared error xiii, xiv, 46, 49, 50, 53–58

MVE mean-variance estimation 2, 15, 19

NGC nerve guidance conduit iv, x, xiii, 9, 10, 32–36, 45, 48, 53, 57

NN neural network xiv, xv, 2, 18, 21, 22, 27, 31, 45, 47, 50, 52–59

OOB out-of-bag 21, 22

PI prediction interval xiv, 14, 15, 17–19, 22, 23, 27, 31, 51–53

PSO particle swarm optimization xiii, xiv, 45–47, 50, 53, 55–58

SWIRL SWarm Intelligence based Reinforcement Learning iv, 45–47, 56–58

B Third-party Libraries

Name:	Log4j 1.2.16
License:	Apache License
Website:	`http://logging.apache.org/log4j/2.x/`
Description:	*"...a popular logging package for Java."*
Scope:	This library was used for all data logging. Software results were logged to later be processed by MATLAB.

Name:	Encog 3.2.0-SNAPSHOT
License:	Apache License
Website:	`http://www.heatonresearch.com/`
Description:	*"...an open source initiative to provide an advanced neural network and bot framework for Java and DotNet."*
Scope:	The Encog Java library is used to build and train all neural networks. Additionally Encog tools are used for data management in particular normalizing data.

Name:	jFuzzyLogic 2.1a
License:	Apache License
Website:	`http://jfuzzylogic.sourceforge.net/`
Description:	*"jFuzzyLogic is a fuzzy logic package written in java (as you might have guessed). It implements Fuzzy control language (FCL) specification (IEC 61131 part 7)"*
Scope:	jFuzzyLogic provides support for the knowledge component of ALINK.

Name:	Commons Configuration 1.6
License:	Apache License
Website:	`http://commons.apache.org/proper/commons-configuration/`
Description:	*"The Commons Configuration software library provides a generic configuration interface which enables a Java application to read configuration data from a variety of sources."*
Scope:	Commons Configuration is used to maintain configuration properties for running the ALINK toolbox. All configuration properties used for data management, model construction, training and testing are accessed using this library.

Name:	Commons Math 3.0
License:	Apache License
Website:	`http://commons.apache.org/proper/commons-math/`
Description:	*"Commons Math is a library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language or Commons Lang."*
Scope:	The Commons Math library is used to perform statistical sampling. Custom U-quadratic distribution was created to generate synthetic training examples to test the functionality of PI implementation.

References

[1] Yaser S Abu-Mostafa. Hints. *Neural Computation*, 7(4):639–671, 1995.

[2] YS Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 1990.

[3] Herman J Ader, Gideon J Mellenbergh, and David J Hand. *Advising on Research Methods: a consultant companion*. Johannes van Kessel Publ., 2008.

[4] P Aebischer, AN Salessiotis, and SR Winn. Basic fibroblast growth factor released from synthetic guidance channels facilitates peripheral nerve regeneration across long nerve gaps. *Journal of neuroscience research*, 23(3):282–289, 2004.

[5] AD Ansselin, T Fink, and DF Davey. Peripheral nerve regeneration through nerve guides seeded with adult schwann cells. *Neuropathology and applied neurobiology*, 23(5):387–398, 1997.

[6] Ran Avnimelech and Nathan Intrator. Boosted mixture of experts: an ensemble learning scheme. *Neural computation*, 11(2):483–497, 1999.

[7] Jeffrey E Barrick, Dong Su Yu, Sung Ho Yoon, Haeyoung Jeong, Tae Kwang Oh, Dominique Schneider, Richard E Lenski, and Jihyun F Kim. Genome evolution and adaptation in a long-term experiment with escherichia coli. *Nature*, 461(7268):1243–1247, 2009.

[8] William G Baxt and Halbert White. Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acute myocardial infarction. *Neural Computation*, 7(3):624–638, 1995.

[9] TB Bini, Shujun Gao, Shu Wang, Aymeric Lim, Lim Ben Hai, S Ramakrishna, et al. Electrospun poly (l-lactide-co-glycolide) biodegradable polymer nanofibre tubes for peripheral nerve regeneration. *Nanotechnology*, 15(11):1459, 2004.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[11] Leo Breiman. Out-of-bag estimation. Technical report, Citeseer, 1996.

[12] Lila J Chamberlain, Ioannis V Yannas, Hu-Ping Hsu, and Myron Spector. Connective tissue response to tubular implants for peripheral nerve regeneration: the role of myofibroblasts. *The Journal of comparative neurology*, 417(4):415–430, 2000.

[13] LJ Chamberlain, IV Yannas, HP Hsu, G Strichartz, and M Spector. Collagen-gag substrate enhances the quality of nerve regeneration through collagen tubes up to level of autograft. *Experimental neurology*, 154(2):315–329, 1998.

[14] Chun-Jung Chen, Yen-Chuan Ou, Su-Lan Liao, Wen-Ying Chen, Shih-Yun Chen, Ching-Wen Wu, Chun-Chiang Wang, Wen-Yi Wang, Yong-San Huang, and Shan-Hui Hsu. Transplantation of bone marrow stromal cells for peripheral nerve repair. *Experimental neurology*, 204(1):443–453, 2007.

[15] Hou-Yu Chiang, Hsiung-Fei Chien, Hsin-Hsin Shen, Jean-Dean Yang, Yu-Hua Chen, Jui-Hsiang Chen, and Sung-Tsang Hsieh. Reinnervation of muscular targets by nerve regeneration through guidance conduits. *Journal of Neuropathology & Experimental Neurology*, 64(7):576–587, 2005.

[16] Pablo Cingolani and Jesus Alcala-Fdez. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[17] Matthew Conforth, Yan Meng, Chandra Valmikinathan, and Xiaojun Yu. Nerve graft selection for peripheral nerve regeneration using neural networks trained by a hybrid aco/pso method. In *Computational Intelligence in Bioinformatics and Computational Biology, 2009. CIBCB'09. IEEE Symposium on*, pages 208–214. IEEE, 2009.

[18] Ralph de Boer, Andrew M Knight, Andreas Borntraeger, Marie-Noëlle Hébert-Blouin, Robert J Spinner, Martijn JA Malessy, Michael J Yaszemski, and Anthony J Windebank. Rat sciatic nerve repair with a poly-lactic-co-glycolic acid scaffold and nerve growth factor releasing microspheres. *Microsurgery*, 31(4):293–302, 2011.

[19] Richard D De Veaux, Jennifer Schumi, Jason Schweinsberg, and Lyle H Ungar. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[20] Wilfred FA Den Dunnen, Berend van der Lei, Jeff M Schakenraad, Engbert H Blaauw, Ietse Stokroos, Albert J Pennings, and Peter H Robinson. Long-term evaluation of nerve regeneration in a biodegradable nerve guide. *Microsurgery*, 14(8):508–515, 2005.

[21] Mahesh Chandra Dodla and Ravi V Bellamkonda. Differences between the effect of anisotropic and isotropic laminin and nerve growth factor presenting scaffolds on nerve regeneration across long peripheral nerve gaps. *Biomaterials*, 29(1):33–46, 2008.

[22] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[23] Richard Dybowski and S Roberts. Confidence intervals and prediction intervals for feed-forward neural networks. *Clinical applications of artificial neural networks*, pages 298–326, 2001.

[24] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*, volume 57. Chapman & Hall/CRC, 1994.

[25] Gregory RD Evans, Keith Brandt, Andreas D Niederbichler, Priscilla Chauvin, Sonja Hermann, Melissa Bogle, Lisa Otta, Bao Wang, and Charles W Patrick. Clinical long-term in vivo evaluation of poly (l-lactic acid) porous conduits for peripheral nerve regeneration. *Journal of Biomaterials Science, Polymer Edition*, 11(8):869–878, 2000.

[26] E.a. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, September 2009.

[27] Paul M George, Rajiv Saigal, Michael W Lawlor, Michael J Moore, David A LaVan, Robert P Marini, Martin Selig, Melvin Makhni, Jason A Burdick, Robert Langer, et al. Three-dimensional conductive constructs for nerve regeneration. *Journal of Biomedical Materials Research Part A*, 91(2):519–527, 2009.

[28] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[29] Sherif Hashem, Bruce Schmeiser, and Yuehwern Yih. Optimal linear combinations of neural networks: an overview. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 3, pages 1507–1512. IEEE, 1994.

[30] J. Heaton. *Introduction to Neural Networks for Java (2nd Edition)*. Heaton Research, 2008.

[31] Tom Heskes. Practical confidence and prediction intervals. *Advances in neural information processing systems*, 1997.

[32] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75, 1994.

[33] James P Hollowell, Aida Villadiego, and Keith M Rich. Sciatic nerve regeneration across gaps within silicone chambers: long-term effects of ngf and consideration of axonal branching. *Experimental neurology*, 110(1):45–51, 1990.

[34] Shan-hui Hsu, Chien-Hsiang Su, and Ing-Ming Chiu. A novel approach to align adult neural stem cells on micropatterned conduits for peripheral nerve regeneration: a feasibility study. *Artificial Organs*, 33(1):26–35, 2009.

[35] JT Gene Hwang and A Adam Ding. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[36] Ryosuke Ikeguchi, Ryosuke Kakinoki, Taiichi Matsumoto, Tomoyuki Yamakawa, Ken Nakayama, Yoshihide Morimoto, Hiroshi Tsuji, Junzo Ishikawa, and Takashi Nakamura. Basic fibroblast growth factor promotes nerve regeneration in a c–ion-implanted silicon chamber. *Brain research*, 1090(1):51–57, 2006.

[37] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[38] C-B Jenq and RE Coggeshall. Numbers of regenerating axons in parent and tributary peripheral nerves in the rat. *Brain research*, 326(1):27–40, 1985.

[39] Chung-Bii Jenq and Richard E Coggeshall. Permeable tubes increase the length of the gap that regenerating axons can span. *Brain research*, 408(1):239–242, 1987.

[40] Chung-Bii Jenq, Lee Lan Jenq, and Richard E Coggeshall. Nerve regeneration changes with filters of different pore size. *Experimental neurology*, 97(3):662–671, 1987.

[41] Yaochu Jin and B Sendhoff. Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, 10(3):231–242, 1999.

[42] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[43] N Kasabov. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 31(6):902–18, January 2001.

[44] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[45] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances.

IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 22(9):1341–56, September 2011.

[46] William Koch, Yan Meng, Munish Shah, Wei Chang, and Xiaojun Yu. Knowledge assistance for uncertain neural networks predicting nerve guidance conduit performance for peripheral nerve injury. *In preparation.*

[47] William Koch, Yan Meng, Munish Shah, Wei Chang, and Xiaojun Yu. Predicting nerve guidance conduit performance for peripheral nerve regeneration using bootstrap aggregated neural networks. In *Neural Networks (IJCNN), The 2013 International Joint Conference on.*

[48] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[49] Annie C Lee, Vivian M Yu, James B Lowe, Michael J Brenner, Daniel A Hunter, Susan E Mackinnon, and Shelly E Sakiyama-Elbert. Controlled release of nerve growth factor enhances sciatic nerve regeneration. *Experimental neurology*, 184(1):295–303, 2003.

[50] Wenbin Li and Mario Fritz. Recognizing materials from virtual examples. In *Computer Vision–ECCV 2012*, pages 345–358. Springer, 2012.

[51] G Lundborg, L Dahlin, D Dohi, M Kanje, and N Terada. A new type of bioartificial nerve graft for bridging extended defects in nerves. *The Journal of Hand Surgery: British & European Volume*, 22(3):299–303, 1997.

[52] R Mart and A El-Fallahi. Multilayer neural networks: an experimental evaluation of on-line training methods. *Computers & Operations Research*, 2004.

[53] D.A. Nix and A.S. Weigend. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 1, pages 55–60 vol.1, 1994.

[54] P Niyogi, F Girosi, and T Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, pages 1–36, 1998.

[55] Partha Niyogi, Federico Girosi, and Tomaso Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 86(11), 1998.

[56] GW Oehlert. A note on the delta method. *The American Statistician*, 46(1):27–29, 1992.

[57] Katsumi Ohbayashi, Hiroshi K Inoue, Akira Awaya, Satoshi Kobayashi, Hideaki Kohga, Masaru Nakamura, and Chihiro Ohye. Peripheral nerve regeneration in a silicone tube: effect of collagen sponge prosthesis, laminin, and pyrimidine compound administration. *Neurologia medico-chirurgica*, 36(7):428, 1996.

[58] Thomas Oommen, Debasmita Misra, Navin KC Twarakavi, Anupma Prakash, Bhaskar Sahoo, and Sukumar Bandopadhyay. An objective analysis of support vector machine based classification for remote sensing. *Mathematical Geosciences*, 40(4):409–424, 2008.

[59] Kevin M Passino and Stephen Yurkovich. *Fuzzy control*. Citeseer, 1998.

[60] Thomas Nordahl Petersen, Claus Lundegaard, Morten Nielsen, Henrik Bohr, Jakob Bohr, Søren Brunak, Garry P Gippert, and Ole Lund. Prediction of

protein secondary structure at 80% accuracy. *Proteins: Structure, Function, and Bioinformatics*, 41(1):17–20, 2000.

[61] DA Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, pages 1–10, 1991.

[62] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document, 1989.

[63] WC Priedhorsky and SS Holt. Long-term cycles in cosmic x-ray sources. *Space Science Reviews*, 45(3-4):291–348, 1987.

[64] Brandon Rasmussen, J Wesley Hines, and Andrei V Gribok. An applied comparison of the prediction intervals of common empirical modeling strategies. In *Proc. Maintenance and Reliability Conference, Knoxville, TN*. Citeseer, 2003.

[65] Paul E Rasmussen, Keith WT Goulding, James R Brown, Peter R Grace, H Henry Janzen, and Martin Körschens. Long-term agroecosystem experiments: assessing agricultural sustainability and global change. *Science*, 282(5390):893–896, 1998.

[66] Joseph Sill and Yaser S Abu-Mostafa. Monotonicity hints. *Advances in neural information processing systems*, pages 634–640, 1997.

[67] Mark Henry Spilker. *Peripheral nerve regeneration through tubular devices: a comparison of assays of device effectiveness*. PhD thesis, Massachusetts Institute of Technology, 2000.

[68] Steven C Suddarth and Alistair DC Holden. Symbolic-neural systems and the use of hints for developing complex systems. *International Journal of Man-Machine Studies*, 35(3):291–311, 1991.

[69] Robert Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, pages 1–15, 1996.

[70] Tamás Varga and Horst Bunke. Generation of synthetic training data for an hmm-based handwriting recognition system. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 618–622. IEEE, 2003.

[71] Lawrence R Williams, Nils Danielsen, Harald Müller, and Silvio Varon. Exogenous matrix precursors promote functional nerve regeneration across a 15-mm gap within a silicone chamber in the rat. *The Journal of comparative neurology*, 264(2):284–290, 2004.

[72] RJ Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, pages 1–10, 1989.

[73] Matthew D Wood, Daniel Hunter, Susan E Mackinnon, and Shelly E Sakiyama-Elbert. Heparin-binding-affinity-based delivery systems releasing nerve growth factor enhance sciatic nerve regeneration. *Journal of Biomaterials Science, Polymer Edition*, 21(6-7):771–787, 2010.

[74] Matthew D Wood, Amy M Moore, Daniel A Hunter, Sami Tuffaha, Gregory H Borschel, Susan E Mackinnon, and Shelly E Sakiyama-Elbert. Affinity-based release of glial-derived neurotrophic factor from fibrin matrices enhances sciatic nerve regeneration. *Acta biomaterialia*, 5(4):959–968, 2009.

[75] Xiaoyun Xu, Hanry Yu, Shujun Gao, Hai-Quan Mao, Kam W Leong, and Shu Wang. Polyphosphoester microspheres for sustained release of biologically active nerve growth factor. *Biomaterials*, 23(17):3765–3772, 2002.

[76] Ioannis V Yannas and Brook J Hill. Selection of biomaterials for peripheral nerve regeneration using data from the nerve chamber model. *Biomaterials*, 25(9):1593–1600, 2004.

[77] Li Yao, Godard CW de Ruiter, Huan Wang, Andrew M Knight, Robert J Spinner, Michael J Yaszemski, Anthony J Windebank, and Abhay Pandit. Controlling dispersion of axonal regeneration using a multichannel collagen nerve conduit. *Biomaterials*, 31(22):5789–5797, 2010.

[78] T Yu, Tony Jan, S Simoff, and John Debenham. Incorporating prior domain knowledge into inductive machine learning. pages 1–42, 2007.

[79] Xiaojun Yu and Ravi V Bellamkonda. Tissue-engineered scaffolds are effective alternatives to autografts for bridging peripheral nerve gaps. *Tissue engineering*, 9(3):421–430, 2003.

[80] Jie Zhang. Developing robust non-linear models through bootstrap aggregated neural networks. *Neurocomputing*, 25(1-3):93–113, April 1999.