Neural Network Evaluation

Michael Gircys

***Abstract*—The efforts outlined in this paper attempt to replicate and evaluate the classification abilities feed-forward neural networks on pair of varyingly difficult datasets. Various neural network configurations are explored, with a focus on comparing and contrasting classification effectiveness between key learning rules, tunable rates, activations functions, and validation methods.**

I. Introduction

THE PURPOSE of these experiments was to evaluate the classification performance of a set of common feed-forward neural network variants across a number of configuration changes. Each of a pair of data sets from the UCI Machine Learning Repository are used to train the neural network instances across a spectrum of tunable configuration rates. Additional changes will be evaluated on the variants and data sets for different activation functions, validation methods, and network topologies.

This report will continue with a review of feed-forward neural networks, the variants employed, and details of the configuration parameters considered for performance evaluation. Following the neural network review will be an overview and description of the analytical methods used to compare classification performance, and the tools used to evaluate confidence in these comparisons. Descriptions of each individual experiment will then be provided, accompanied with the corresponding results, and discussion points found from the data. Finally, we will summarize the results found and conclude the experiments.

As neural networks are well suited to classification tasks, and the focus of our evaluation is performance on classifications, we will require training data with some known properties that will allow us to observe a performance baseline. The data sets used to evaluate the networks were obtained from the UCI Machine Learning Repository [2], and include the Iris Plant Type [1] patterns set and the Breast Cancer Wisconsin (Diagnostic) [6] set. These sets are ideal for classification, and are accompanied by attribute and class information, as well as known solvers where available.

The first set - Iris Plant Type [1] - provides 150 samples with 3 classifications (one linearly separable) from 4 real-type attributes. Iris plant subspecies is to be determined from 4 measurable metrics of the flower. The data provides a simple domain for classification, and should be a suitable test set to generate baseline configurations. We may expect quicker convergence, and a preference for simpler topologies. One experiment performed involves determining the effectiveness of holdout folds on generalisation ability, and may benefit from a larger amount of samples than was provided. To this effect, all experiments make use of a data set which triples the count of original samples by applying an inverse normal distribution on all values, with the original value as the mean and a 0.1 variance.

Fig. 1. Iris Plant Type Data Definition

Iris Plant Type		
Attributes	Sepal length	(cm)
	Sepal width	(cm)
	Petal length	(cm)
	Petal width	(cm)
Classifications	Iris Setosa	
	Iris Versicolour	
	Iris Virginica	

Fig. 2. Breast Cancer Wisconsin (Diagnostic) Data Definition

Breast Cancer Wisconsin (Diagnostic)		
Attributes	Clump Thickness	1 - 10
	Uniformity of Cell Size	1 - 10
	Uniformity of Cell Shape	1 - 10
	Marginal Adhesion	1 - 10
	Single Epithelial Cell Size	1 - 10
	Bare Nuclei	1 - 10
	Bland Chromatin	1 - 10
	Normal Nucleoli	1 - 10
	Mitoses	1 - 10
Classifications	Benign	
	Malignant	

The Breast Cancer Wisconsin (Diagnostic) [6] set provides a slight increase in classification difficulty. We are provided with 699 samples with 2 classifications from 10 real-type attributes. A sample is determined to be benign or malignant based on measured cell properties and behaviours. While the number of classifications has been reduced, it does not appear that any solution has been previously found which was capable of perfectly separating the classes. A more complex set of data relating to properties of 3 specific cells per sample is available, but is outside of the scope of these experiments.

II. Neural Networks

A neural network is a supervised machine learning algorithm ideal for classification problems. A directed graph is produced by generating layers of nodes, and providing a weight to each edge connecting them. Each node weighs and sums the data passed to it before evaluating the result on its node-specific activator function. This function output can then be passed to any nodes which can be visited from the current node. By providing attribute samples with known,

expected results, the network can self-adjust and attempt to minimize produced errors. The neural networks employed for the experiments in this report are feed-forward neural network variants.

Feed-forward neural networks constitute one of the simpler families of neural network topologies, where the data between each layer of neurons is directed in a single direction - from input layer to output layer. The neurons within each layer of the network are typically fully connected to every neuron in its neighbouring layers. Extending from the simplest perceptron topology, a feed-forward neural layer may include multiple hidden layers, each with a variable amount of hidden nodes. By permitting additional layers and intermediate nodes, we are able to generate increasingly complex function compositions, which allows us to produce correspondingly complex classification logic.

A. Variant: BackProp

Neural network backwards propagation of error (or back-propagation, or BackProp) is the fundamental method of distributing output layer error measures to the inner neurons of the network. In training with back-propagation, we are able to compute the actual output of the model with the associated input and compare to the expected output. The difference between expected and actual output can provide the error value for each neuron in the output layer. We are then able to iteratively assign error values to the neurons in each layer moving backwards. Each neuron is assigned its error of by summing the errors in all the neurons in the layer ahead of it, factored by the amount its output contributed to those neurons (weight).

With each neuron assigned an error value, a determination of how much each weight within the neuron must change can be made. For each weight, the derivative of the activation function can be used in conjunction with the total neuron error, total weighted neuron input, and individual input value (that value which gets adjusted by the current weight) to find the error gradient for the weight. This gradient reflects the amount that the error will change based on changes to the weight, assuming all other weights are equal. The calculation of the neuron errors and weight gradients is outline in Algorithm 1. The weights are then updated by this gradient (factored by the tunable learning rate), and momentum (, the amount of change from the previous iteration, discussed below).

As the other variants employed require a batch learning approach, the variant of BackProp that was used for analysis within this report also made us of a batch learning methodology. The changes from the traditional BackProp method are minimal; the gradient value is simply averaged across all of the training values, and the final weight update is performed at the end of each epoch. This results in less error noise, and ensures that each sample is evaluated from the same state within an epoch instead of being influenced by the samples evaluated before it. The final BackProp batch algorithm is outline in Algorithm 2.

Algorithm 1 Error Gradient Calculation

$weight[layer, neuron, weight]$
$state[layer, neuron]$
$error[layer, neuron]$

for all $\{inputs, expected\} \in data\_samples$ **do**
  $state \Leftarrow Evaluate(inputs)$
  $output \Leftarrow state_{outputlayer}$

  {Move backwards and determine proportional error}
  for $i = 0$ to $|outputs|$ **do**
    $error_{outputlayer,i} \Leftarrow expected_i - output_i$
  end for
  for $l = outputlayer - 1$ to $inputlayer + 1$ **do**
    for $n = 1$ to $|layer.neurons|$ where $layer = l$ **do**
      $error_{l,n} \Leftarrow \sum_w error_{l+1,w} \times weight_{l+1,n,w}$
      {inputs provided which produced the error}
      $sum \Leftarrow \sum_w state_{l-1,w} \times weight_{l,n,w}$
      for $w = 1$ to $|neuron.weights|$ where $neuron = n$ **do**
        $gradient_{l,n,w} \Leftarrow \frac{\partial f_{l,n}(sum)}{\partial sum} \times error_{l,n} \times state_{l-1,w}$
      end for
    end for
  end for

end for

Algorithm 2 BackProp

η {learning rate}
ζ {momentum rate}

for all $\{i, w, m\} \in weights$ **do**
  i {weight index}
  w {weight value}
  m {momentum; previous weight change}
  $E_t \Leftarrow compute\_gradients(i)$
  $E_t = -\sum_s \frac{\partial E_s}{\partial \omega_{i,s}}(t) = -\sum_s \delta_{i,s} \frac{\partial f_n(e_s)}{\partial e_s} x_{i,s}$
  {sum of gradients for a weight across all batch samples}

  $\Delta \Leftarrow \eta E_t + \zeta m$
  $w \Leftarrow w + \Delta$
  $m \Leftarrow \Delta$
end for

B. Variant: RProp

RProp is a neural net batch learning method dependant on the same error gradient calculation as vanilla back-propagation. However, while the weight update method is dependant on this error calculation, it uses only the sign of the gradient, not the magnitude. Instead of relying on value of the error gradient to determine the weight change delta, RProp keeps track of an update value for each weight, which grows or shrinks by empirically established values based on the error gradient maintaining its current sign. It was found that a growth rate of 1.2, and shrink rate of 0.5 gave optimal

results [5]. The update value is typically also clamped to [0.000001,50] for similar motivations as moving away from gradient magnitude [5]. Algorithm 3 displays the method of weight and update value adjustment in RProp.

RProp has been found to perform well for many applications, and converges more quickly that vanilla BackProp. By maintaining its own update value per weight, learning rates and momentum rates become redundant. The method in which the update value is adjusted appears similar to a high momentum BackProp model. This has the benefit of allowing for operation without the need for additional parameter tuning, but despite the momentum-like behaviour, tends to converge to the first minima.

Algorithm 3 RProp

$\eta^+ \Leftarrow 1.2$
$\eta^- \Leftarrow 0.5$
$\Delta_{max} \Leftarrow 50.0$
$\Delta_{min} \Leftarrow 0.000001$

for all $\{i, w, dw, d, E_{t-1}\} \in weights$ **do**
  i {weight index}
  w {weight value}
  dw {previous weight change, signed}
  d {weight delta}
  E_{t-1} {previous gradient}
  $E_t \Leftarrow compute\_gradients(i)$
  $E_t = -\sum_s \frac{\partial E_s}{\partial \omega_{i,s}}(t) = -\sum_s \delta_{i,s} \frac{\partial f_n(e_s)}{\partial e_s} x_{i,s}$
  {sum of gradients for a weight across all batch samples}

  if $sign(E_{t-1}) = 0$ **then**
    $dw \Leftarrow d \times sign(E_t)$
  else if $sign(E_{t-1}) = sign(E_t)$ **then**
    $d \Leftarrow min(d \times \eta^+, \Delta_{max})$
    $dw \Leftarrow d \times sign(E_t)$
  else $\{sign(E_{t-1}) \neq sign(E_t)\}$
    $d \Leftarrow max(d \times \eta^-, \Delta_{min})$
    $E_t \Leftarrow 0$
  end if

  $w \Leftarrow w - dw$
  $E_{t-1} \Leftarrow E_t$
end for

C. Variant: Delta-Bar-Delta

The Delta-Bar-Delta (DBD) algorithm appears not dissimilar to a sub-variant of RProp. Instead of mimicking momentum by adjusting the magnitude of a weight delta, we instead directly shrink or grow a dynamic learning rate.

Each weight value is updated in a similar manner to that of traditional BackProp; weights are incremented by the error gradient value, factored by a learning rate. However, we must first update the learning rate that will be used. Based on whether or not the sign of the error gradient has changed, the learning rate associated with a given weight grows or decays. We can increment the learning rate by a fixed growth value K, or decrement the rate by a factor of $(1 - D)$ for decay value D in $(0, 1)$. This provides a sort of meta-learning adjustment, increasing learning when it appears to be improving the model, and decreasing learning when it begins to go astray [4]. Tunable options may include the initial learning rate, growth value, and decay percentage.

D. Other Configurables

A number of other attributes and learning rules can be configured, and may provide measurable performance gains or losses. This report will experiment with learning rate, momentum rate, hidden layer and node set-up, activation function selection, and holdout methods.

1) Learning Rate: The neural network variants update the weights associated with each nodes inputs based on an amount of error found to come from the given node. However, a weight value which causes error for one input may be required for providing a correct output for some other input. Continuous training may lead to oscillation of weights based on the competition produced from the two inputs. To help allow the network to converge, a learning rate may be employed, scaling back the amount of effect each update iteration will have on the node weights. The value of the learning rate is expected to be in $(0, 1]$, and is a tunable parameter with which we will experiment.

2) Momentum Rate: Conversely, to avoid early convergence on local extrema, the concept of momentum with a tunable momentum rate can be used. By tracking the change in each weight from the previous iteration, this amount of change can be applied to the current iteration as well. This allows a consistently improving weight to continue to update in the same direction despite a minor amount of iterations where the weight would otherwise be adjusted opposite its last direction. The hope is that this momentum value will allow the weight to pass local error minima, where a stronger maxima will sufficiently influence the momentum to change direction. Similar to the learning rate, a tunable $[0, 1]$ momentum rate interval will undergo experimentation.

3) Hidden Nodes: A simple neural net perceptron has only its input and output layers to train. The feed-forward networks used in these experiments permit additional layers which should be considered hidden from any program using the implementation. With additional nodes between the input and output layers, an increasingly complex function composition can be produced, potentially allowing correspondingly intricate classification logic. The number of additional layers and the number of nodes within each of these layers may be configured at initialization time, likely with great influence on classification runtime performance and accuracy.

4) Activation Function: While evaluating an input within a neural network, one of the critical steps performed in each neuron is to pass the weighted sum of its inputs through an activation function. Traditionally, this function is the logistic (or soft-step, or sigmoid) function, producing a $(0, 1)$ output to the next node. However, other functions can also be used as an activator, for a single neuron, or the entire network. In comparison to the identity function, other functions were

found to be interesting based on how they scaled and smoothed inputs. As the neural network variants employed in the experiments are all dependant and error gradient optimization, one key requirement for the activation functions is that there must exist a 1st order derivative for the activator. While not required, it was found that non-linear [3], monotonic [11], smooth [15] activators seemed to perform most reliably. Within our planned experiments, we will evaluate how a neural network completely configured with the TanH function (giving $(-1, 1)$) performs in comparison to the sigmoid function.

5) Validation Techniques: A concern with supervised learning methods is over-fitting the training data. Over-fitting occurs when the model begins to conform to specific data samples instead of providing a more general abstraction of the structures and correlations within the data. Where no effort is made to control this, models may exhibit excellent performance on the training data, but fail to provide proper results on new data samples. One common remedy is to partition the data samples into training and validation sets, where training is on only performed on the training data set. The validation set can then be evaluated, where a drastically lower score would suggest over-fitting. If over-fitting is detected in the way, it may be sufficient to terminate the run early before the model is further compromised, however noise can make this difficult to reliably detect [8].

An alternate method of maintaining generalization may be to further abstract the data partitioning into a larger amount of partitions, known as folds. Multiple instances of the neural net can be created, where each train on all but one of the data sets. The remaining data set is used for validation. These sets could have been previously partitioned into a global testing set for a more accurate test performance. The best performing of the folds can be used for final validation and testing. This would require a suitably large original data set to ensure a sufficient count of samples in folds' validation sets. It was for this reason that the Iris data was tripled by adding noise to the original samples.

III. Methods of Analysis

Within the analysis of our experiments, we will measure performance through the means of the MSE across the testing sample set. Mean Square Error (MSE) provides an interpretable measure of error distance across numerous sample dimensions, and has useful known properties. MSE happens to be an approximation for the error variance across the testing set, which will assist in a number of the statistical tests that we will perform [13]. When displaying performance of experiment configurations which span multiple runs, the mean of these values across all the runs will be used.

To further evaluate and measure the strength of any comparisons, a number of statistical displays and tests will be used. We will briefly discuss confusion matrices, confidence intervals, T-tests, and ANOVA measures.

A. Confusion Matrix

While generally quite useful and easy to interpret, averages and other aggregates based on MSE may not be the most accurate display of true classification performance. The output values for a successful (or failed) classification should be found after a threshold to above or below 0.5 (ie. more confidence that an input gives one class than not). This suggests that a system with a higher MSE may classify more accurately than a system with a lower MSE, so long as the first system has values closer to - though still on the correct side of - the appropriate thresholds. This may fundamentally be unsolvable, as any data set may have noise within the measured attributes, and it is not guaranteed that all classifications will be linearly separable. Without the use of validation measures, any classification with extremely low training error should be viewed with suspicion of over-fitting.

One way that we can attempt to avoid the actual classification error vs mean square error ambiguity is to employ confusion matrices. In supervised learning methods, where the expected outputs are known, it should be trivial to track whether or not an input gives a certain class, and whether or not is was expected to give that class. Where an input is found to have the class that it should, or doesn't have a class which it shouldn't, we have a true positive or true negative respectively. Conversely, by having a class which it shouldn't, or not having a class that it should, the input provides a false positive or false negative respectively. By tracking these true/false positives/negatives, we can find interesting derived error measurements.

The accuracy of a classification, measuring how often the classifier is actually correct, can be found from $(TP + TN)/Total$. Its opposite, the error rate, is simply $(FP + FN)/Total$. With additional information from the input and output values, The true positive rate (TP/TP_{max}), and false positive rate (FP/TN_{max}) can be determined, giving how often the classifier gives a class when the sample should give the class, and how often the classifier gives a class when the sample shouldn't give the class. In a similar manner, specificity refers to how often the classifier withholds a class when it should, and precision refers to the percentage of correct positive predications. Where the threshold can be varied, an ROC curve plots the true positive rate against the false positive rate as the threshold is scaled. [16]

For classifications with more than two possible classes, the confusion matrix can be extended to track all possible predicated classifications for all actual classifications. For the purpose of determining specificity within activation function bias, we can group all possible classification in the same binary confusion matrix - correctly or incorrectly assigned or didn't assign a particular classification. The experiments in this test assume a fixed threshold of 0.5, over or under which a class will or will not be assigned.

B. Confidence Interval

Confidence intervals and confidence intervals go hand-in-hand. A confidence interval of 95% would suggest that in repeating an experiment, we should find our resultant samples should fall within our original range for 95% of our repeated tests. This original range is our confidence interval. Thus, in some given example where there exists 95% confidence of

a value being between x and y, that there is a 95% chance (confidence level) that the value will indeed be within $[x, y]$ (confidence interval).

Without knowing additional information about the sample population, we assume that it follows a t-distribution. Properties known about this t-distribution allow us to determine an interval about a mean, for given sample attributes and a desired confidence. For convenience, t-distribution is sufficiently well studied, and lookup tables are available. The formulae below [10] permits us to determine such a confidence range with use of a lookup table:

$$d_f = |samples| - 1$$

$$\alpha = \frac{1}{2}(1 - conf_{level})$$

$$t = t_{table}(d_f, \alpha)$$

$$d = t \times \frac{\sigma_{samples}}{\sqrt{|samples|}}$$

$$conf_{interval} = [\mu_{samples} - d, \mu_{samples} + d]$$

We would expect, with a given confidence level, that future repetitions of a given experiment will provides values which lie within this found range.

C. T-Test

The unnamed Student's T-test is able to determine the confidence that a pair of samples have a different mean. It may be seemingly apparent through visual analysis that some given group has a notably different performance than some other group across a number of runs. However, this is not always the case. These visual interpretations may only be based on mean performance, where a high deviation may suggest that the way in which the two distributions overlap is not as clear. An insufficient sample size could also remove confidence in any comparative result. A T-test is able to provide a confidence metric for the likelihood of distribution overlap based on mean, standard deviation, and sample size.

The calculations within a t-test are similar to that of signal-to-noise ratio evaluation. We must find the difference in the two sets' means in proportion to the two groups' variability.

$$t = \frac{|\mu_a - \mu_b|}{\sqrt{\frac{\sigma_a}{n_a} + \frac{\sigma_b}{n_b}}}$$

$$d_f = (n_a - 1) + (n_b - 1)$$

With the t-value and degrees of freedom calculated as above, the same t-distribution table used to determine confidence intervals can be referenced to determine significance value of the samples having different means [9].

D. ANOVA

A T-test is useful for comparison of two distributions. However, for comparison of an arbitrary number of sample sets, a one-way analysis of variance (ANOVA) test should be used first. An ANOVA can be used to determine if there is any significance between the means of multiple experimental groups. One advantage in the use of an ANOVA over multiple T-tests is the reduction of error; each T-test inherently induces a small chance of suggesting a false positive, which stacks cumulatively with each additional testing using its source data. A one-way ANOVA on only two distributions is identical to the confidence measure obtained from a T-test [9].

An ANOVA does require a number of assumptions to be made, namely that dependant variables are normally distributed in each group, the sample variances in each group are roughly equal, and observations are independent [14]. The one-way ANOVA is considered to be relatively robust against non-normal distributions, though particularly flat distributions should be remedied through transforms before use in the test. The ANOVA determines and evaluates an F-test statistic against a known table of significance values for the hypothesis that the the sample distributions are equal. To find the F-statistic, we can apply the following formula:

$$F = \frac{explained\ variance}{unexplained\ variance}$$

$$F = \Sigma_{i=1..k} \frac{n_i(\mu_i - \mu)^2}{k - 1}$$

Generalizing from the one-way ANOVA, a two-way anova can be used to compare the effect of multiple variables on resultant distributions. Where the one-way ANOVA tests the hypothesis that the two distributions are equal across all measures of a variable, the two-way ANOVA must also test that the distributions are equal across all measures of the second variable, and optionally that the two variables are independent.

IV. Results and Discussion

The base configuration that will be used in the experiments, unless otherwise noted, are listed in Table I. These configuration parameters were determined in part through best practices, and from the initial experiments on the BackProp learning and momentum rates. While a run count of 6 could be increased for additional confidence, concessions were originally made to accommodate for practical run time - particularly in the case of the Breast Cancer Wisconsin (Diagnostic) data set. The limit on the number of Epochs was also set with a consideration for run time limitations. An implementation problem resulted in a substantially increased runtime while processing the cancer data set, and while the problem was eventually corrected and the experiment re-run, the previous epoch and run count is maintained for consistency. This count appeared to be generally suitable to display most convergence behaviours. Where explicitly required, the epoch limit has been increased and noted in the experimental parameters.

TABLE I
EXPERIMENTAL BASELINE CONFIGURATION

Parameter	Iris	Cancer
Runs	6	6
Epochs	500	500
Topology	4-4-3	9-9-2
Activation Function	Sigmoid	Sigmoid
Learning Rate	0.6	0.8
Momentum Rate	0.3	0.3
Training Samples	70%	70%
Testing Samples	30%	30%
Holdout Folds	1	1

A. Variant: BackProp

The following experiments evaluate BackProp performance from tunable configuration changes and use of the original BackProp learning method in batch mode.

TABLE II
EXPERIMENTAL CONFIGURATION - BACKPROP RATES

Parameter	Iris	Cancer
Runs	10	10

Group	Learning Rate	Momentum Rate
01	0.2	0.0
02	0.4	0.0
03	0.6	0.0
04	0.8	0.0
05	0.2	0.3
06	0.4	0.3
07	0.6	0.3
08	0.8	0.3
09	0.2	0.6
10	0.4	0.6
11	0.6	0.6
12	0.8	0.6

TABLE III
TWO-WAY ANOVA SUMMARY - BACKPROP RATES - IRIS

Source	SS	d_f	MS	F	P
Momentum Rate	0.02	2	0.01	4.15	0.0183
Learning Rate	0.02	3	0.01	2.77	0.0451
Interaction	0.02	6	0	1.38	0.2292
Error	0.26	108	0		
Total	0.32	119			

TABLE IV
TWO-WAY ANOVA SUMMARY - BACKPROP RATES - CANCER

Source	SS	d_f	MS	F	P
Momentum Rate	0.00	2	0	0	1
Learning Rate	0.00	3	0	0	1
Interaction	0.01	6	0	4.5	0.0004
Error	0.04	108	0		
Total	0.05	119			

1) Rates: In evaluating the effect of the rates for a neural network operating on the iris data set, a two-way ANOVA shows some notable relation between performance and learning rate, momentum, and a combination of the two. A low learning rate appeared to have performance plateau at a higher level of error, though with that low learning rate, performance increased slightly with an elevated momentum rate. A marginal improvement was seen from the group with the highest learning rate, though the best results for the set were had from combinations of mid level learning and momentum rates. The optimal pairing for the set was had with a learning rate of 0.6, and a momentum rate of 0.3. Running a t-test on this optimal configuration against the other configurations gives a P value of 0.052, just reaching the 95% confidence level, suggesting that the ideal configuration is statistically significant.

The two-way ANOVA on the cancer set shows that after 500 epochs, there was not a particularly strong relation between performance and learning rate nor between performance and momentum rate as applied to the set. However, there was a definite effect on performance within the interaction of the two rates. In comparison to the iris set, performance appeared to plateau at a more consistent level, though different speeds of convergence can be seen around epoch 100.

2) Hidden Nodes: A critical experiment that was performed measured the performance change after adjusting the network topology. For the cancer and iris data sets, the baseline configuration relies on a single hidden layer, where the number of nodes is identical to the number of input nodes. To contrast this, we evaluate networks where the number of hidden nodes is halved or doubled. Additionally, instead of just doubling the number of hidden nodes in a single layer, an additional experiment is performed where the entire hidden layer is duplicated.

TABLE V
EXPERIMENTAL CONFIGURATION - BACKPROP TOPOLOGIES

Group	Iris Set	Cancer Set
01	4-2-3	9-5-2
02	4-4-3	9-9-2
03	4-4-4-3	9-9-9-2
04	4-8-3	9-18-2

TABLE VI
ONE-WAY ANOVA SUMMARY - BACKPROP TOPOLOGY - IRIS

Source	SS	d_f	MS	F	P
Treatment	0.043	3	0.014	13.0	<0.0001
Error	0.022	20	0.001		
Total	0.066	23			

TABLE VII
ONE-WAY ANOVA SUMMARY - BACKPROP TOPOLOGY - CANCER

Source	SS	d_f	MS	F	P
Treatment	0.007	3	0.002	6.87	0.0023
Error	0.007	20	0.001		
Total	0.013	23			

The iris set's 4-4-4-3 neural network appeared to improve more rapidly than the 4-2-3 or 4-4-3, but then quickly plateaued at around epoch 200, as seen in figure 5. However, The 4-4-3 network appeared to still be gradually improving after 500 epochs. A one-way ANOVA confirms significant relation in performance to topology. A clear victor, the 4-8-3 network showed substantial performance gains over the other networks. While the 4-8-3 network appeared to have an increase in noise and lower stability at the higher epoch count, a t-test validates strong confidence in this better-performing configuration. While a higher number of hidden nodes lead

Fig. 3. Effects of Learning Rate and Momentum Rate on Iris Data Set

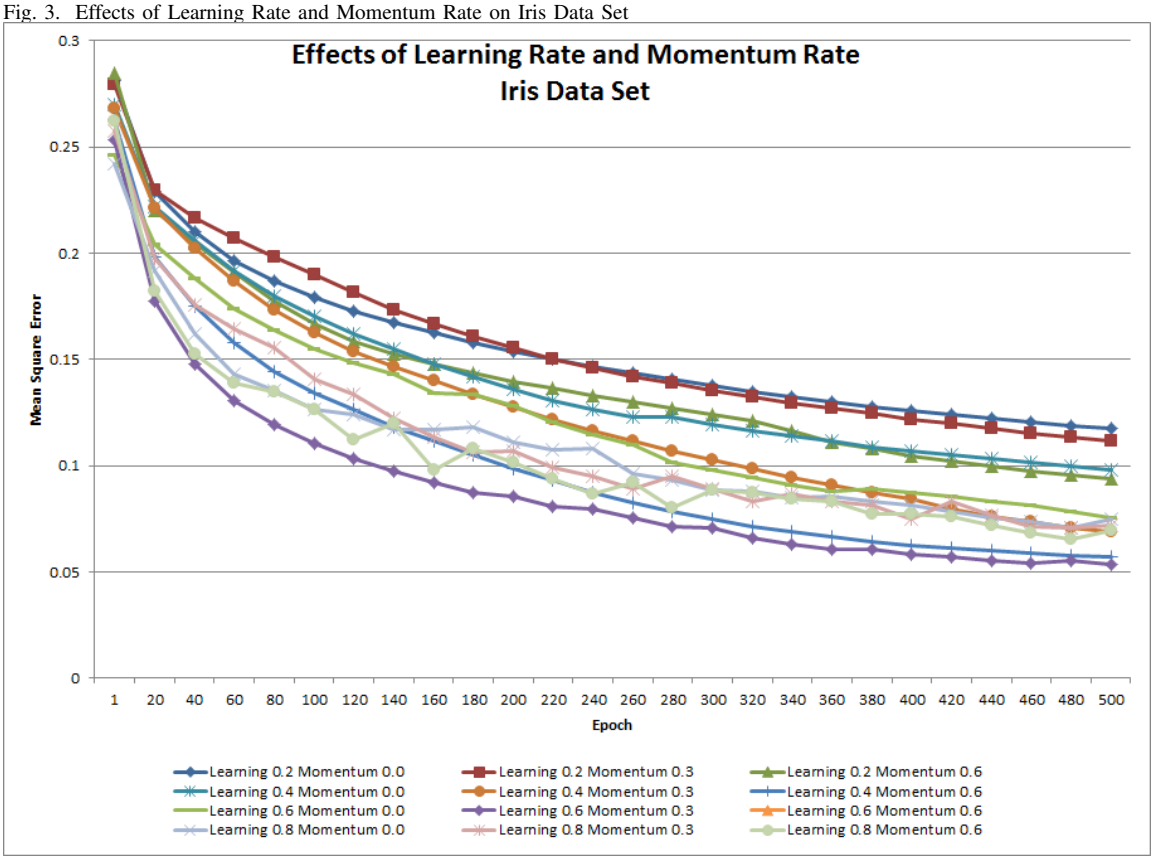


Fig. 4. Effects of Learning Rate and Momentum Rate on Cancer Data Set

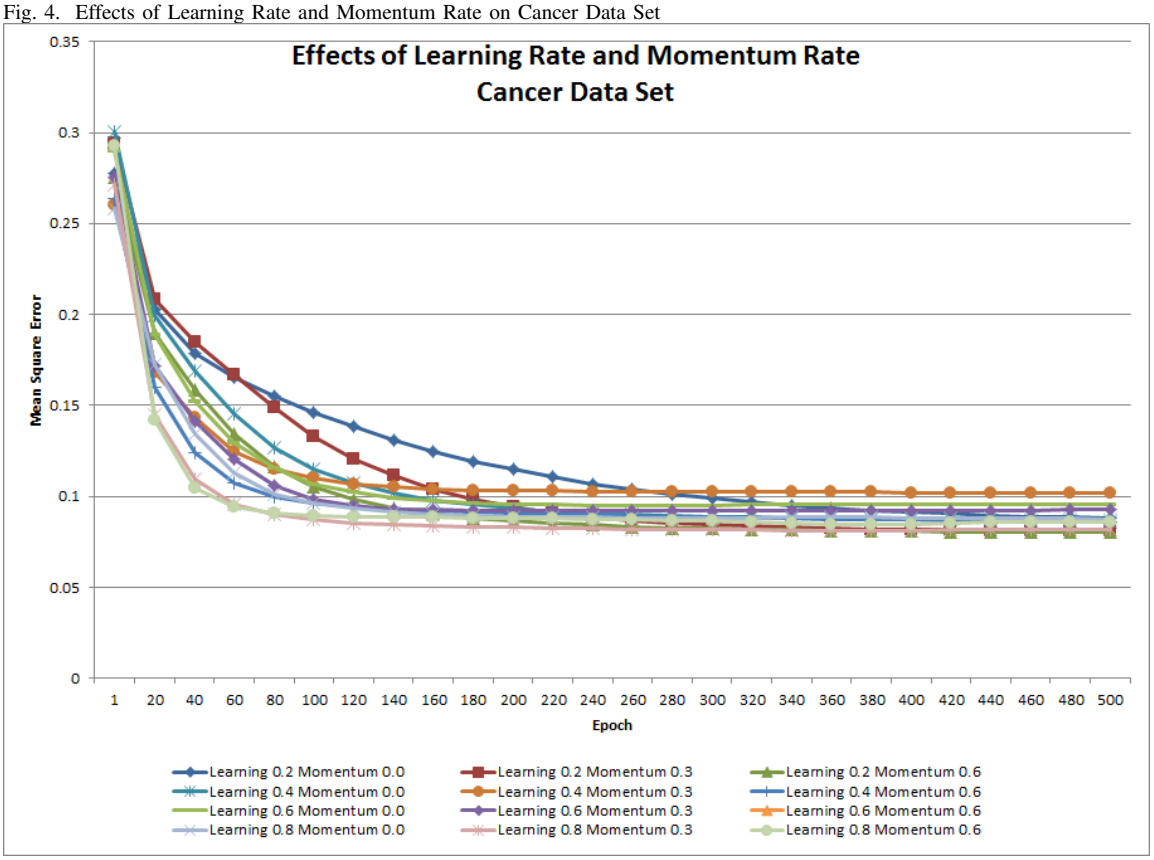


Fig. 5. BackProp - Effects of Hidden Nodes - Iris Data Set

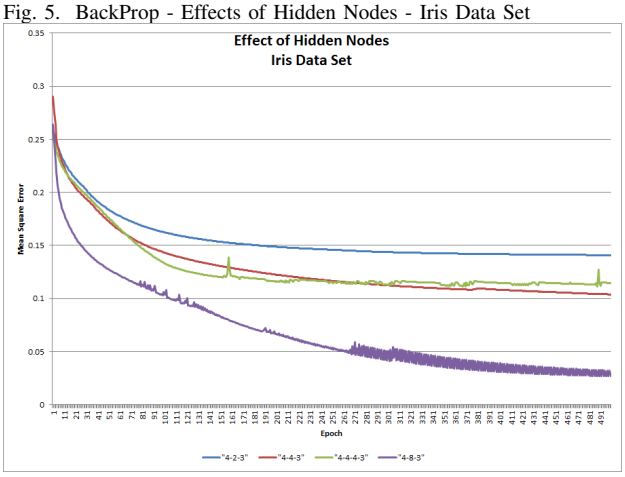
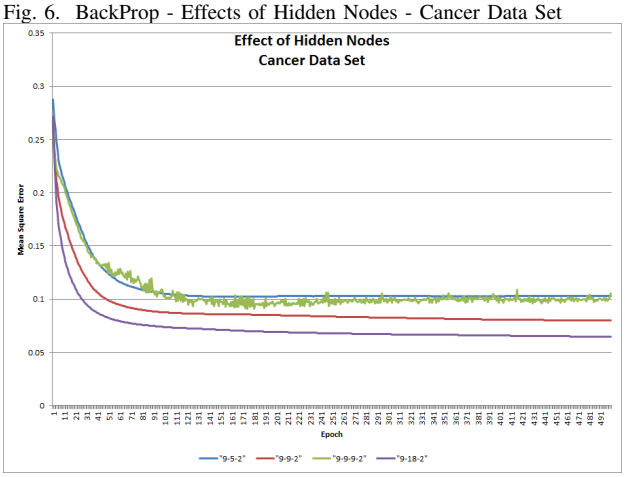


Fig. 6. BackProp - Effects of Hidden Nodes - Cancer Data Set



to the optimal found configuration, it is curious as to why a similar number of nodes across multiple hidden layers lead to such early, poor convergence.

The cancer set (figure 6) shows a similar effect, showing with strong confidence results which mirror those had on the cancer set. While the difference in performance levels is not as great, mirrored in our reduced ANOVA P-value, we still show a strong 99% confidence level in benefit of a hidden layer with count double that of the input layer. Similar to the plateau seen with the iris set's 4-4-4-3 network, the 9-9-9-2 network run on the cancer data set shows a similar plateau effect, and begins to later increase in error. It seems likely that the additional layer encourages overfitting of the training data.

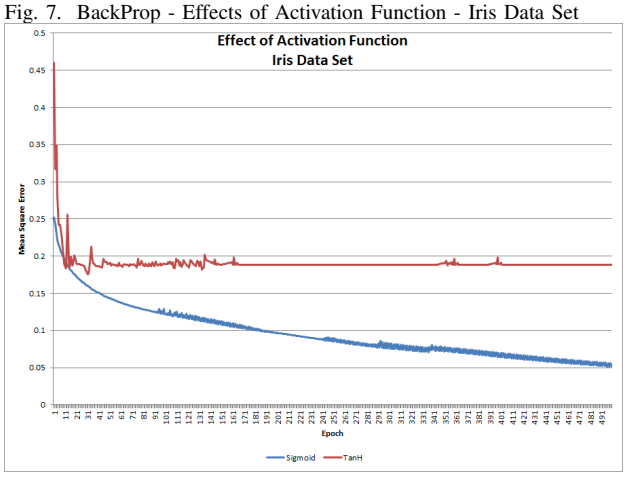
3) Activator Functions: While the number of potential activations is quite large, two of the more frequently encountered neuron activation functions are the Sigmoid (or, logistic), and Tanh functions [12]. We will compare neural networks where every neuron within them uniformly uses a single activator function. Blurb about activation functions.

It seems immediately clear from figures 7 and 8 that the tanh function did not perform nearly as well as the sigmoid function in BackProp for either of the data sets. An initial error may have been expected, due to the increased distance to a

TABLE VIII
EXPERIMENTAL CONFIGURATION - BACKPROP ACTIVATION FUNCTIONS

Group	Activation Function
01	Sigmoid
02	TanH

Fig. 7. BackProp - Effects of Activation Function - Iris Data Set



possible expected 1.0 output from an initial -1.0 (contrasted in sigmoid with 0.0). Despite accepting a higher initial error, the network with the iris data was unable to overcome an initial learning plateau. T-tests between functions on the two sets gives strongly significant P-values of 0.002 and 0.007, despite the heavy noise seen in the charted averages.

The tanh function has a $(-1, 1)$ range centred about 0.0, as opposed to the sigmoid's $(0, 1)$ range about 0.5. As the classification threshold has been fixed at 0.5, it would seem likely that an adjustment to the threshold may be required to improve performance while using tanh as an activation function. The requirement of a threshold adjustment is undermined somewhat by a weaker, or positive effect of the activation function with other neural network variants such as delta-bar-delta (as seen in figure 20). To examine the effect of the threshold, we will examine the false negative rates of the aggregate confusion matrices.

From the confusion matrices in table IX, we can see an

Fig. 8. BackProp - Effects of Activation Function - Cancer Data Set

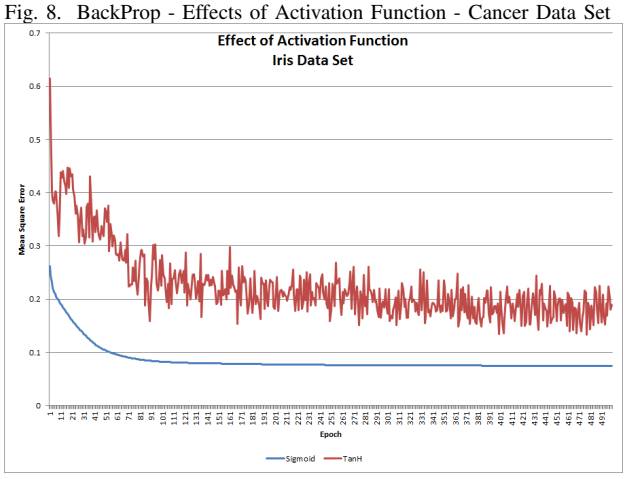


TABLE IX
CONFUSION MATRICES - BACKPROP ACTIVATION FUNCTIONS

Iris Set - Sigmoid

	T	F
P	30.2%	03.1%
N	63.5%	03.1%

Iris Set - Tanh

	T	F
P	10.4%	04.7%
N	61.9%	22.9%

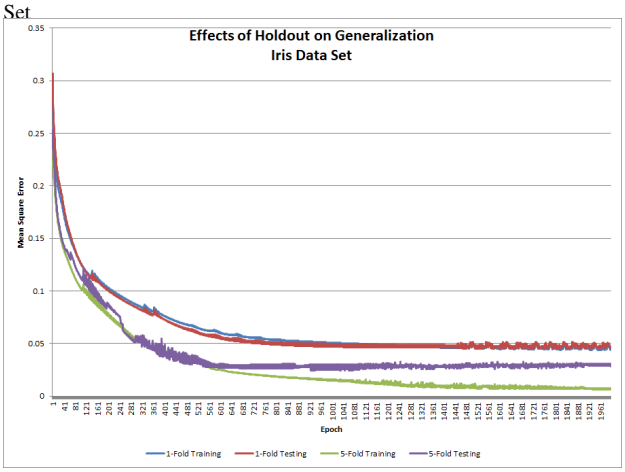
Iris Set - Sigmoid

	T	F
P	47.0%	03.0%
N	47.0%	03.0%

Iris Set - Tanh

	T	F
P	47.0%	13.6%
N	36.4%	03.9%

Fig. 9. BackProp - Effects of Holdout on Generalization Ability - Iris Data Set



elevated amount of false negative for the tanh function on the iris data set, but the hypothesis that the threshold should be adjusted may be rejected after observation of an opposite trend in the cancer data set. With the Breast Cancer Wisconsin (Diagnostic) set, we actually see a stable false negative count, but elevated false positive count in comparison to the sigmoid runs when using tanh. While the cause is not immediately clear, it would appear that the use of the tanh function may encourage an early plateau in classification accuracy, possibly with heaving oscillation.

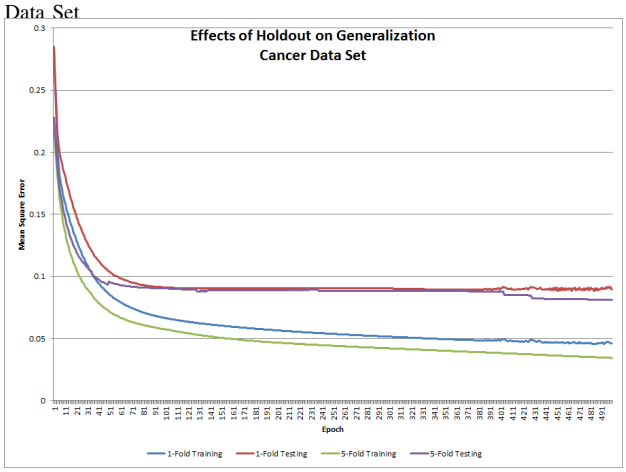
4) Validation Methods: One concerning observation that was made during previous experiments in the report was overfitting of the training data which was marked by a raising test error measure. Holdout is a critical technique that allows a performance measure of the network in a more general case. An additional technique often employed to maintain high performance while minimizing overfitting is a k-fold cross validation. We will compare both the testing and training performance scores across a configuration using only a single fold, and a configuration with 5 folds.

TABLE X
EXPERIMENTAL CONFIGURATION - BACKPROP VALIDATION METHODS

Group	Validation
01	1-fold with No Validation (Training Only)
02	1-fold with 30% Testing Holdout
03	5-fold with No Validation (Training Only)
04	5-fold with 30% Testing Holdout

One thing that is clear from the graphs in figures 9 and 10 is that the testing error, after sufficient training time, will flatten

Fig. 10. BackProp - Effects of Holdout on Generalization Ability - Cancer Data Set



or even begin to raise as training performance improves. The other problematic observation is in the noise and intersection of training and testing data during the runs with the iris data set. As confirmed from [8], short term characteristics of the training and testing performance may not be sufficient to determine an optimal stopping criteria.

An unexpected observation was that in both the iris and data cancer sets, a significant classification performance gain on the testing measures was found by using 5-fold cross validation. It was initially expected that 5-fold cross validation may permit a longer training duration before testing performance converged, but it would appear that we had also achieved a quicker rate of performance. This performance may only be superficial, as the improvement to the iris set and cancer set hold P-values of 0.33 and 0.44 respectively, which do not suggest any strong significance at the current sample size. One hypothesis may be that the additional folds, and thus additional models to train, increased the likelihood of beginning with a better initial network state. As the best performing model within the folds is reported on, this skews the performance distribution in its favour. The actual computing cost/benefit ratio may not be reflected with this scheme.

B. Variant: RProp

The following experiments evaluate RProp performance from tunable configuration changes.

1) Rates: Both a benefit and limitation of RProp is the lack of necessary tunable rates. The method in which the RProp update value is adjusted performs similar function to that of the momentum concept. Further, as the sign of the error gradient affects the learning behaviour, but not the magnitude, any static learning rate factor is necessarily useless. RProp does use static values for the positive and negative update value adjustments, but these values have been empirically determined to perform best at 1.2 and 0.5 respectively [5].

2) Hidden Nodes: The experiments with RProp use the baseline configuration as with BackProp, with the obvious exception of the learning methodology. Consequently, for comparative analysis to BackProp, we will evaluate the per-

Fig. 11. RProp - Effects of Hidden Nodes - Iris Data Set

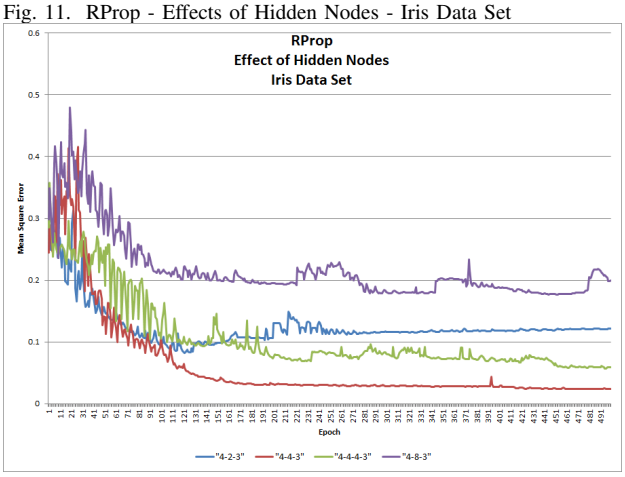
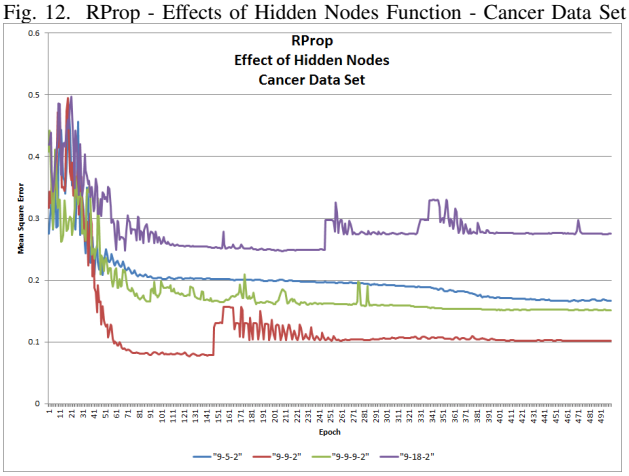


Fig. 12. RProp - Effects of Hidden Nodes Function - Cancer Data Set



formance of RProp using the same groups of hidden node variations.

TABLE XI
EXPERIMENTAL CONFIGURATION - RPROP TOPOLOGIES

Group	Iris Set	Cancer Set
01	4-2-3	9-5-2
02	4-4-3	9-9-2
03	4-4-4-3	9-9-9-2
04	4-8-3	9-18-2

For all the various topologies used with RProp, performance measures seemed to display a much larger amount of noise or oscillation. This appears to be a consistent divergence from BackProp based on the other experiments performed with RProp, and may be a notable quirk of the algorithm (or perhaps the specific implementation).

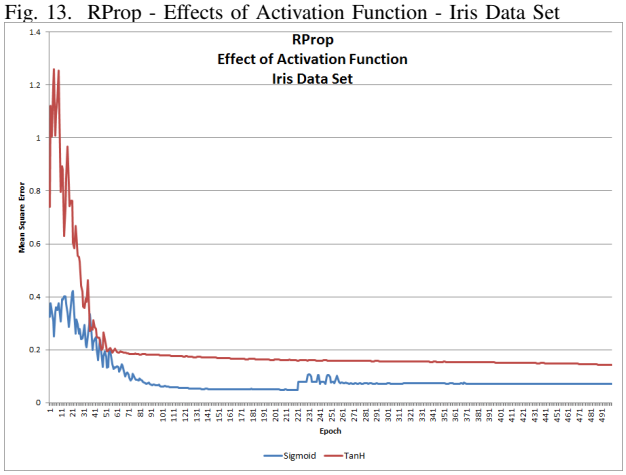
TABLE XII
ONE-WAY ANOVA SUMMARY - RPROP TOPOLOGY - IRIS

Source	SS	d_f	MS	F	P
Treatment	0.107	3	0.036	2.2	0.120
Error	0.324	20	0.016		
Total	0.431	23			

TABLE XIII
ONE-WAY ANOVA SUMMARY - RPROP TOPOLOGY - CANCER

Source	SS	d_f	MS	F	P
Treatment	0.096	3	0.032	1.57	0.228
Error	0.406	20	0.020		
Total	0.502	23			

Fig. 13. RProp - Effects of Activation Function - Iris Data Set



The high level of noise provides difficulty in performing analysis at the early epochs, and any significance of the analysis will be correspondingly reduced. Similar to BackProp, the groups with reduced hidden nodes appear to flatten in the middle epochs, though the cancer set does show some minor improvements continue. In a complete contrast to BackProp, where the 4-8-3 and 9-18-2 networks ranked best, the same topologies appeared to rank worst with RProp. With the noise on the graphs and the small sample size, the ANOVA P-values are consequently reduced to 0.12 and 0.23 which do not present the strongest confidence in the correlation. T-test evaluations between the Iris 4-4-4 and the baseline confirm this, though a 95% confidence level was found in the case of the Cancer set's 9-18-2 layout. In both the cancer and iris data set evaluations, the single hidden layer with double neuron count group had a marked increase in errors which should signal significant overfitting as a prime culprit for the poor performance.

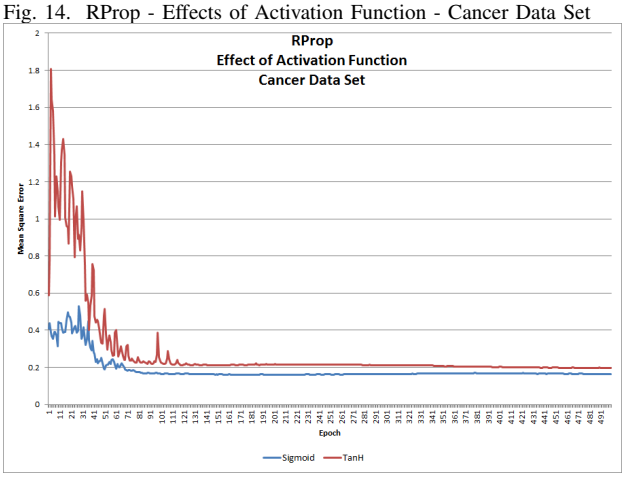
3) Activator Functions: With the elevated noise seen the previous experiment, concerns were present as to whether the previously oscillating tanh function would exacerbate the issue. Conversely, while high noise is seen in the early epochs, error rates appears to flatten out and converge at around the 100th epoch for both data sets (see figures 13 and 14).

TABLE XIV
EXPERIMENTAL CONFIGURATION - RPROP ACTIVATION FUNCTIONS

Group	Activation Function
01	Sigmoid
02	TanH

In further contrast to the results found with BackProp, the performance measures between sigmoid and tanh across both data sets converged to similar levels. The first 50 epochs with tanh did show a behaviour common to the BackProp

Fig. 14. RProp - Effects of Activation Function - Cancer Data Set



experiments, which was a notably higher initial level of error. RProp appears to be capable of bypassing the early error minima that was encountered with BackProp; the momentum-like behaviour of RProp may be more suitable than the baseline BackProp parameters when making use of the tanh activation function.

TABLE XV
CONFUSION MATRICES - RPROP ACTIVATION FUNCTIONS

Iris Set - Sigmoid

	T	F
P	28.0%	03.5%
N	63.2%	05.3%

Iris Set - Tanh

	T	F
P	18.4%	05.8%
N	60.9%	14.9%

Cancer Set - Sigmoid

	T	F
P	49.5%	11.4%
N	38.6%	00.5%

Cancer Set - Tanh

	T	F
P	37.9%	12.6%
N	37.4%	12.1%

In continuing the hypothesis that tanh introduces thresholding range issues, we can view the confusion matrices for the RProp activation function experiments. Both iris and cancer data sets display a minor rise in false positive count, and a substantial rise in false negative count. As both data sets have mutually exclusive classifications, we should expect an optimal true negative rate of 66.6% for the iris data set and 50.0% for the cancer set. Where sigmoid shows a slightly higher than expected false positive count, the tanh experiments on RProp show a slightly elevated count of false positives in comparison to the expected count.

C. Variant: Delta-Bar-Delta

The following experiments evaluate Delta-Bar-Delta performance from tunable configuration changes.

1) Rates: Where RProp had empirically optimal tunings for how much its delta value was adjusted, no such measures have been provided for the Delta-Bar-Delta growth and decay rates. Sutton [4] has offered rates of 0.02 and 0.20 for growth and decay respectively, and it is around these values that we will experiment. Table XVI gives the listing of growth and decay rate pairs evaluated.

With the performances of the groups being closely intertwined, it was unknown if any clear associations with the

TABLE XVI
EXPERIMENTAL CONFIGURATION - DELTA-BAR-DELTA RATES

Group	Growth Rate	Decay Rate
01	0.02	10%
02	0.02	30%
03	0.10	10%
04	0.10	30%

Fig. 15. Delta-Bar-Delta - Effects of Growth/Decay Rates - Iris Data Set

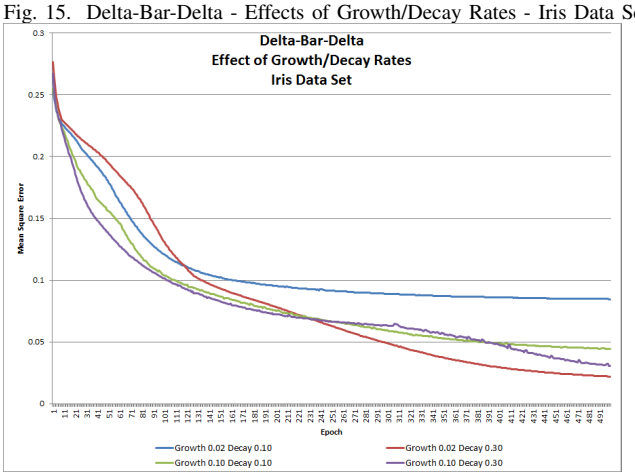


TABLE XVII
TWO-WAY ANOVA SUMMARY - DELTA-BAR-DELTA RATES - IRIS

Source	SS	d_f	MS	F	P
Decay Rate	0.01	1	0.01	10.0	0.0049
Growth Rate	0	1	0	0	1
Interaction	0	1	0	0	1
Error	0.02	20	0		
Total	0.03	23			

TABLE XVIII
TWO-WAY ANOVA SUMMARY - DELTA-BAR-DELTA RATES - CANCER

Source	SS	d_f	MS	F	P
Decay Rate	0	1	0	0	1
Growth Rate	0	1	0	0	1
Interaction	0	1	0	0	1
Error	0.01	20	0		
Total	0.01	23			

Fig. 16. Delta-Bar-Delta - Effects of Growth/Decay Rates - Cancer Data Set

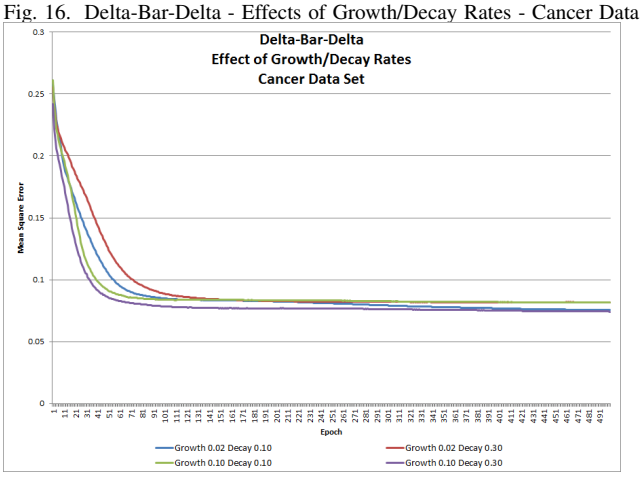


Fig. 17. Delta-Bar-Delta - Effects of Hidden Nodes - Iris Data Set

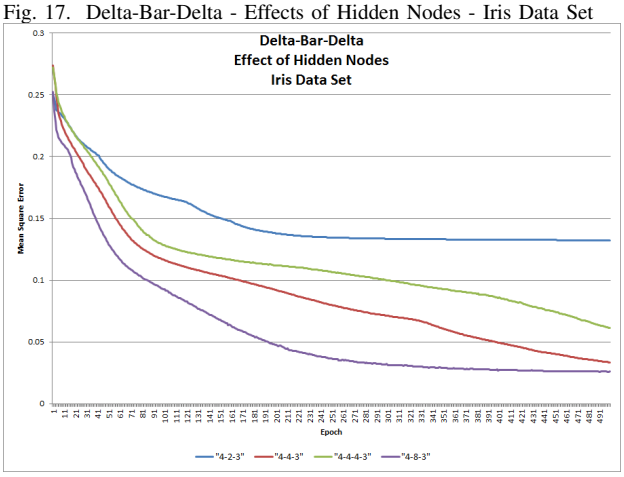
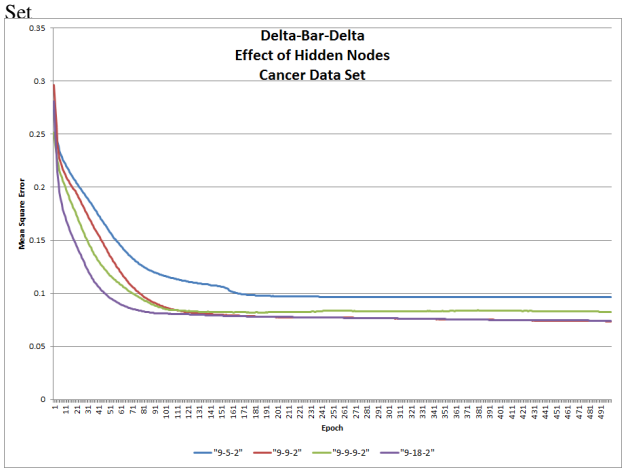


Fig. 18. Delta-Bar-Delta - Effects of Hidden Nodes Function - Cancer Data Set



rates and performance could be found. Upon completion of a two-way ANOVA for both data sets, one significant link was found. The growth and decay rates did not appear to correlate to any effect on performance with the Breast Cancer Wisconsin (Diagnostic) data set. However, the iris data set had a high significance correlation with the decay rate. The best two results both utilized the higher decay rate of 0.3 . The growth rate was still found to be without significance for the iris data set with the baseline run count.

2) Hidden Nodes: In continuing a comparative analysis to BackProp and RProp, we will evaluate the performance of Delta-Bar-Delta learning methods using the same groups of hidden node variations. In contrast to RProp, each of the various topologies provide smooth performance curves with clear separation (or coincidence) between the groups at early and late epochs.

TABLE XIX
EXPERIMENTAL CONFIGURATION - DELTA-BAR-DELTA TOPOLOGIES

Group	Iris Set	Cancer Set
01	4-2-3	9-5-2
02	4-4-3	9-9-2
03	4-4-4-3	9-9-9-2
04	4-8-3	9-18-2

Consistent with the results found in the BackProp and RProp experiments, the groups with reduced hidden nodes flatten in the middle epochs. Delta-Bar-Delta appears to agree with BackProp in that the 4-8-3 and 9-18-2 layer structures provide the best performance. However, the 4-4-4-3 and 9-9-9-2 structures give mediocre performance - not immediately appearing to match the excellent RProp effectiveness nor performing as poorly as the BackProp run with the same topologies. The 9-9-9-2 layout did see some minor overfitting and flattening at epoch 200.

TABLE XX
ONE-WAY ANOVA SUMMARY - DELTA-BAR-DELTA TOPOLOGY - IRIS

Source	SS	d_f	MS	F	P
Treatment	0.042	3	0.014	20.8	<0.0001
Error	0.014	20	0.001		
Total	0.056	23			

TABLE XXI
ONE-WAY ANOVA SUMMARY - DELTA-BAR-DELTA TOPOLOGY - CANCER

Source	SS	d_f	MS	F	P
Treatment	0.002	3	0.001	1.51	0.243
Error	0.009	20	0.001		
Total	0.011	23			

The two-way ANOVA for the hidden layer options on the cancer data set showed a very mild significance, but the performance measure for the iris data set were held with high statistical confidence. For RProp, our baseline performed admirably, with 4-8-3 still best, and 4-2-3 worst. With the increasing improvements on the 4-4-4-3 data set in the last epochs of the cancer data set evaluation, it could be interesting to re-run the experiment for additional training iterations - figure 17 may lead one to believe that 4-4-3 or 4-4-4-3 could overtake the flattening 4-8-3 due to their increased performance in the last 100 epochs.

3) Activator Functions: The sigmoid and tanh functions are lastly evaluated for both the iris and cancer data sets using a delta-bar-delta learning variant. While initial error is high, error appears to follow a smoother curve in comparison to RProp or BackProp, though some oscillation is still present in the iris set after flattening out at epoch 200.

TABLE XXII
EXPERIMENTAL CONFIGURATION - DELTA-BAR-DELTA ACTIVATION FUNCTIONS

Group	Activation Function
01	Sigmoid
02	TanH

The tanh function finally appears capable of competing with sigmoid. Despite starting at a much larger error, tanh presented on the iris data set the ability to improve at a quicker rate than sigmoid, and surpassed sigmoid's performance for 200 epochs ending at around epoch 250 when performance with tanh levelled off. With the cancer data set, tanh did not improve at a significantly quicker rate than sigmoid, but was able to continue improving at a slightly better rate at their intersection in epoch 300. T-tests showed little significance in

Fig. 19. Delta-Bar-Delta - Effects of Activation Function - Iris Data Set

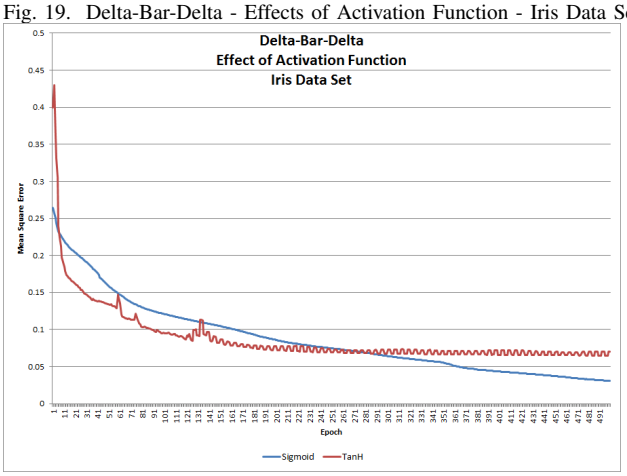
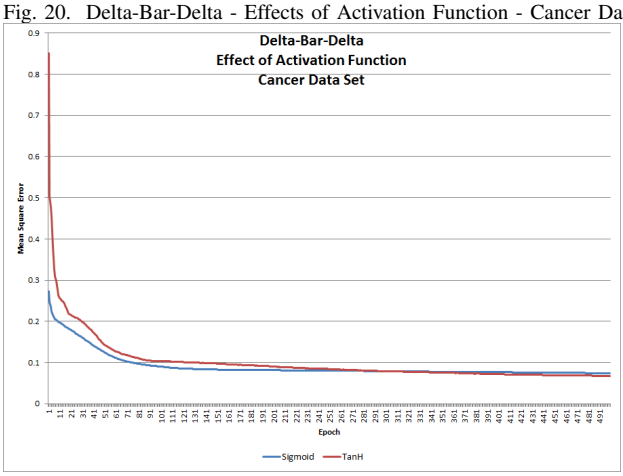


Fig. 20. Delta-Bar-Delta - Effects of Activation Function - Cancer Data Set



the results. While not an objective success for the function in these applications, performance equality with sigmoid is the best result the tanh activator function has received across the neural network variants tested.

TABLE XXIII
CONFUSION MATRICES - DELTA-BAR-DELTA ACTIVATION FUNCTIONS

Iris Set - Sigmoid

	T	F
P	32.4%	00.7%
N	66.0%	00.9%

Iris Set - Tanh

	T	F
P	29.4%	04.2%
N	62.5%	03.9%

Cancer Set - Sigmoid

	T	F
P	48.2%	01.8%
N	48.2%	01.8%

Cancer Set - Tanh

	T	F
P	48.7%	02.7%
N	47.3%	01.3%

While the relative equality of the activation functions in their use with the Delta-Bar-Delta learning method might have suggested a lack of noteworthy observations in the confusion matrix, it has been included for posterity. Expectantly, the confusion matrix results offer no meaningful contributions to the thresholding offset hypothesis.

V. CONCLUSION

The learning rate and momentum rate configurations offered statistically significant performance variances. Best results for both data sets were found with middle-range values for learning rate and momentum rate. By using the principal of momentum in the model, results were notably improved. In the Delta-Bar-Delta learning method with the adaptive learning rate, it was seen that the decay rate could have a significant impact on performance, though the growth rate did not appear to contribute significantly for the data sets used.

Methods to track and negate overfitting - the lack of generalization - are critical for obtaining optimal results with neural networks. Many of the experiments explored in this report produced networks with some minor overfitting. In some cases, such as in figure 12, the training could be terminated much earlier and result in a better performing network. The use of multiple folds did result in slightly better networks, though it was not found to have been with any high statistical significance. Similar performance benefits per computer runtime cost may be found in re-training the same neural network model.

Mixed results were seen in regards to variations in the neural network topologies. Generally, the baseline (which used a single layer of hidden neurons, the number of which was identical to the input layer) performed reasonably well, and halving the number of neurons in the hidden layer gave poor results. Using an increased number of neurons, over 1 or two hidden layers, was somewhat dependant on the learning method to determine if better results could be had. Overfitting seemed likely, and whether or not the network would first achieve low error performance was somewhat inconsistent. With the RProp results giving higher variance, it would seem likely that an increase in neurons in the first neural net layer would be advantageous for the data sets used in the problem.

A consistent advantage was found towards the groups of models using the sigmoid function over tanh. In all but one case, the sigmoid function clearly outperformed the tanh function by decreasing error and reducing noise and variance. At best, for the given data sets, the tanh function provided equal performance to sigmoid. At worst, tanh converged to an error level statistically higher than that of its competitor. For the provided data sets, learning methods, and baseline configurations, the sigmoid function offers increased performance with greater reliability.

While some configuration options had clear victors which clearly outperformed others, not all provided statistically optimal choices. It is possible that an increased run size could provide higher confidence for some of the run results. Additionally, interdependence between other tunable values could be explored.

REFERENCES

[1] Michael Marshall R.A. Fisher. Uic machine learning repository - iris data set. http://archive.ics.uci.edu/ml/datasets/Iris, 1936. [Online; accessed 2016-03-01].

[2] David Aha. Uic machine learning repository. http://archive.ics.uci.edu/ml/datasets.html, 1987. [Online; accessed 2016-03-01].

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. [via Wikipedia].

[4] Richard S Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pages 171–176, 1992.

[5] Martin Riedmiller and I Rprop. Rprop-description and implementation details. 1994.

[6] Nick Street, et al. Uic machine learning repository - breast cancer wisconsin (diagnostic) data set. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+1995. [Online; accessed 2016-03-01].

[7] Tom M Mitchell et al. Machine learning, 1997.

[8] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[9] William M.K. Trochim. The t-test. http://www.socialresearchmethods.net/kb/stat_t.php, 2006. [Online; accessed 2016-03-01].

[10] StatisticsHowTo Stephanie. Confidence interval: How to find a confidence interval: The easy way! http://www.statisticshowto.com/how-to-find-a-confidence-interval/, 2009. [Online; accessed 2016-03-01].

[11] Huaiqin Wu. Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. *Information Sciences*, 179(19):3432–3441, 2009. [via Wikipedia].

[12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[13] David M. Lane. One-factor anova (between subjects). http://onlinestatbook.com/2/analysis_of_variance/one-way.html, 2013. [Online; accessed 2016-03-01].

[14] Laerd Statistics. One-way anova. https://statistics.laerd.com/statistical-guides/one-way-anova-statistical-guide.php, 2013. [Online; accessed 2016-03-01].

[15] Michael S Gashler and Stephen C Ashmore. Training deep fourier neural networks to fit time-series data. In *Intelligent Computing in Bioinformatics*, pages 48–55. Springer, 2014. [via Wikipedia].

[16] Data School. Simple guide to confusion matrix terminology. http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/, 2014. [Online; accessed 2016-03-01].