1999-07-16

The Little Neuron that Could

Timothy L. Andersen

Tony R. Martinez
martinez@cs.byu.edu

Original Publication Citation

Andersen, T. L. and Martinez, T. R., "The Little Neuron that Could", Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99, CD paper #191, 1999.

The Little Neuron that Could

Tim Andersen
tim@axon.cs.byu.edu
Brigham Young University
Provo, Utah

Tony Martinez
martinez@cs.byu.edu
Brigham Young University
Provo, Utah

Abstract

SLPs (single layer perceptrons) often exhibit reasonable generalization performance on many problems of interest. However, due to the well known limitations of SLPs very little effort has been made to improve their performance. This paper proposes a method for improving the performance of SLPs called "wagging" (weight averaging). This method involves training several different SLPs on the same training data, and then averaging their weights to obtain a single SLP. The performance of the wagged SLP is compared with other more complex learning algorithms (bp, c4.5, ib1, MML, etc) on 15 data sets from real world problem domains. Surprisingly, the wagged SLP has better average generalization performance than any of the other learning algorithms on the problems tested. This result is explained and analyzed. The analysis includes looking at the performance characteristics of the standard delta rule training algorithm for SLPs and the correlation between training and test set scores as training progresses.

1. Introduction

For any given *d* dimensional classification problem a single layer perceptron (SLP) is limited to generating a *d*-1 dimensional decision surface (hyperplane). Any problem which can be completely solved by an SLP is therefore referred to as being linearly separable. It is commonly thought that SLPs are not sufficiently powerful to perform well on most types of classification problems [20]. It is difficult to argue with this notion for two reasons. First, the ratio of linearly separable problems to all possible problems quickly approaches zero as the problem dimension increases, which implies that an SLP is only capable of an exact solution on an extremely small subset of the possible problems. Second, most real world problems do not exhibit the characteristic of linear seperability, and these are the problems of primary interest. These two observations have motivated the development of learning algorithms which are capable of generating more complex decision surfaces.

Unfortunately, coupled with an algorithm's ability to generate more complex hypotheses is a higher likelihood that the algorithm will overfit the data. So, utilizing a more complex learning algorithm, while perhaps guaranteeing *the ability* to generate the solution we are looking for, does not necessarily guarantee that it will in fact generate a good solution (one that performs well on unseen data). There are many methods available to help guard against the problem of overfitting [11], [7], all of which are based upon either a complexity-accuracy tradeoff or the use of a holdout set. However, overfitting avoidance is an enormously difficult problem, and no method for guarding against overfitting has been shown to work well on all types of learning problems. The main difficulty lies in the provable fact that it is impossible to determine solely from the data what level of complexity is acceptable [15], [13]. This means that when testing a complex learning algorithm on a large variety of applications there will often be at least a few applications which the algorithm performs poorly on due to overfitting. This tendency for a complex learning algorithm to perform poorly on a few applications tends to counteract any exceptional performance that the algorithm may have on other applications.

This paper tests and analyzes a method for improving the performance of an SLP which we call "wagging" for weight **averaging**. With wagging, two simple modifications are made to the standard perceptron training algorithm which significantly improves the generalization performance of the SLP. The first modification is to save the best weight vector (in terms of training set accuracy) produced during training, and the second is to average the weight vectors obtained from several different training runs. These two modifications result in a 23 percent average improvement in the error rate for an SLP. This improvement is enough to boost the performance of the SLP so that its average test set results are significantly better than several other 'more capable' machine learning and neural network algorithms tested in this paper. The other learning algorithms include an MLP trained with backpropogation, c4.5, id3, ib1, cn2, and others. These other algorithms are briefly described in section 2.

Section 2 introduces the data sets and discusses the various machine learning and neural network algorithms which are used in this paper for comparison purposes. Section 3 explores ways to improve the performance of SLPs. Section 4 looks at the performance characteristics of the standard approach used to train SLPs, and why wagging improves this performance. Discussion of results is given in section 5 and the conclusion is given in section 6.

2. Algorithms and Data

Table 1 lists the data sets used in this paper. The first column gives the name (or tag) used to identify the data set throughout the rest of this paper. The total number of

0-7803-5529-6/99/$10.00

attributes is listed in the third column, and the fourth column gives the total number of examples contained in the data set. These data sets were obtained from the UCI machine learning database repository. All of these data sets are based upon real world problem domains, and are more or less representative of the types of classification problems that occur in the real world.

tag	full name	attributes	instances
bc	breast cancer	9	286
bcw	breast cancer wisconsin	10	699
bupa	bupa liver disorders	7	345
credit	credit approval	15	690
echo	echocardiogram	13	132
sickeu	sick-euthyroid	26	3163
hypoth	hypothyroid	26	3123
ion	ionosphere	35	351
promot	promoter gene sequence	57	106
sick	sick	30	3772
sonar	sonar	61	208
stger	german credit numeric	24	1000
sthear	statlog heart	13	270
tic	tic-tac-toe	9	958
voting	house votes 1984	16	435

Table 1. Data sets.

The scores reported throughout the rest of this paper for the other learning algorithms are taken from [16]. All of the algorithms tested in this paper are trained/tested using 10-fold cross validation on the same data splits that were used in [16].

Tag	Full Name
bp	multilayer perceptron
per	perceptron
c4	c4
c4.5	c4.5
ib1	instance based 1
id3	id3
mml	IND v2.1
smml	IND v2.1
cn2	unordered cn2

Table 2. Learning algorithms.

The learning algorithms we compare against are summarized in table 2. The first column lists the name which is used to refer to the corresponding learning algorithm throughout the rest of this paper. The second column gives the usual name used to refer to the learning algorithm, and the last column lists some references for each learning algorithm.

3 Improving the performance of an SLP

The output z of a perceptron for the kth input pattern is defined as

$$z = \begin{cases} 1 \text{ if } \sum_a w_a x_{ak} + \theta > 0 \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

Where w_a is the weight on input attribute a
x_{ak} is the value of attribute a for example k
θ is an adjustable threshold

The standard training or weight update formula for a perceptron is the well known delta rule ([9]), which is

$$\Delta w_a = c(t_k - z)\, x_{ak} \quad (2)$$

Where c is the learning rate
t_k is the target classification for example k

On many real world data sets an SLP has generalization performance which compares favorably to other, more complex learning algorithms. On the data sets where an SLP does not perform as well as more complex methods, the main problem may not be that an SLP is inherently incapable of comparable generalization performance, but that the training algorithm simply did not pick the best weight vector for generalization. The weight vector generated by one training run can differ significantly from the weight vector from another training run, even if the training accuracies of the two weight vectors are the same. When given several different but equivalently performing (on the training set) weight vectors it is important to determine which one will have the best generalization performance.

3.1 Bagging

One way to circumvent the problem of having to choose the single best weight vector is to train several different SLPs and use them all by having them vote for the output classification. This approach, termed "bagging", has been used with good success [2][5][8][28]. The standard bagging approach is defined as follows. Let B be the number of predictors φ we wish to generate. First, B training sets are formed from the available training set T by taking repeated random samples from T, then each of these training sets T_k is used to train a predictor φ_k. The output o of the aggregation φ_B of all the φ_k predictors on input x is then

$$z = l_j \,|\, \forall i (i \neq j \rightarrow \sum_{k=1}^{B} \delta(l_j, \varphi_k(\mathbf{x})) > \sum_{k=1}^{B} \delta(l_i, \varphi_k(\mathbf{x}))) \quad (3)$$

where l_i is the label for the ith output class
δ is the Kronecker delta function

It can be shown that the aggregate classifier φ_B will tend to have generalization performance which is at least as good as the average performance of all the φ_k, assuming that the φ_k are reasonably good (better than random) classifiers. The key to obtaining actual performance improvements with bagging is the degree of instability in the φ_k. In other words, the φ_k must differ somewhat in the errors they exhibit for φ_B to improve classification over the average classification performance. Randomly permuting the data with the perceptron training technique essentially guarantees that different training runs will produce different solutions with correspondingly different errors. Bagging is therefore a natural fit for dealing with the many different weight vectors that can be produced from multiple training runs on the same data for an SLP.

3.2 Wagging

For the special case of a linear perceptron the output z on input x is defined as

$$z = \sum_i w_i x_i \qquad (4)$$

where x_i is the ith element of x.

Define w_{ki} to be the ith weight of the kth perceptron. The output of an aggregation of B linear perceptrons using bagging is then

$$z = 1/B \sum_k \sum_i w_{ki} x_i = \sum_i \left(\sum_k 1/B\, w_{ki} \right) x_i \qquad (5)$$

Bagging multiple linear perceptrons is therefore equivalent to averaging their weights to obtain a single linear perceptron. We term this approach "wagging" for weight averaging.

Taking the average of all the available weight vectors is also one way to estimate the most probable weight vector, since the average weight vector is a reasonable approximation to the most probable weight vector (given the training algorithm and data) for an SLP. This approach has been tested with multilayer networks and been shown to work well ([14]).

There are two disadvantages to bagging. The main disadvantage of bagging is that it must store and run many different predictors. Wagging solves this problem by requiring that only a single predictor/weight vector, the average, be stored and run. For the bagging approach defined by equation 3 another possible disadvantage of bagging for classification problems is that it does not take into account the confidence of each predictor. With neural networks the activation of the net can loosely be viewed as the confidence that the network has in its prediction. By using a single vote per predictor, it is possible for situations to arise where several weak predictors outvote a few strong ones, which may not be desirable. Wagging could improve generalization performance for bagged SLPs in the case where network activation is correlated with confidence.

3.3 Bagging vs Wagging

Table 3 compares the performance of bagging and wagging on 15 real world data sets. These results are averages obtained using 10-fold cross validation on the available data. For each training set 100 different SLPs were generated by retraining 100 times using random permutations of the training set between each training iteration. Weights were initially set to zero, and the maximum number of training iterations was set at 10,000 due to time constraints. The best weight vector (BWV) was used from each training run as the final weight setting for each SLP. The BWV was determined by pausing at the end of each training iteration and testing the SLP on the entire training set and then saving the highest scoring weight vector. The rationale behind using the BWV rather than the most recent weight vector will be explained in section 4. Equation 3 was used to obtain the estimated output for the bagged SLPs. For the wagged SLP we used equation 6, which is derived from equation 5, to obtain the estimated output of a single SLP with averaged weights.

$$z = \begin{cases} 1 \text{ if } \sum_k \sum_i w_{ki} x_i > 0 \\ 0 \text{ otherwise} \end{cases} \qquad (6)$$

	wag	bag	high	low	avg	t-test
bcw	**97.00**	96.71	98.00	94.57	96.61	0.92
bc	**73.14**	72.80	79.08	61.58	71.47	0.70
bupa	69.26	**69.28**	78.26	57.38	67.61	-0.51
credit	84.93	**85.36**	90.00	79.71	85.19	-0.78
echo	89.34	89.34	94.67	80.16	87.90	-0.50
sickeu	**96.52**	96.36	97.28	94.37	95.95	0.89
hypoth	98.29	98.29	98.96	97.44	98.22	0.50
ion	**88.62**	88.33	94.03	80.08	87.84	0.71
promot	**91.55**	90.55	100.00	60.36	84.51	0.72
sick	96.87	96.87	97.64	95.52	96.69	0.50
sonar	**77.38**	75.90	86.00	64.36	75.42	0.91
stger	**76.40**	75.70	80.50	68.00	74.65	0.88
sthear	83.33	**83.70**	90.00	71.48	82.42	-0.83
tic	**98.33**	98.22	99.06	95.72	97.85	0.83
voting	95.19	95.19	97.02	91.26	94.82	0.82
avg	87.74	87.51	92.03	79.47	86.48	

Table 3. Bagging vs wagging using 100 SLPs.

The column labeled "high" reports the highest test set accuracy (averaged over the 10 cross validations) of the 100 SLPs, the "low" column reports the lowest test set accuracy, and the "avg" column reports the average test set accuracy of the 100 SLPs. The last column gives the confidence using the student t-test that wagging is better than bagging for each data set. If bagging is better than wagging then the confidence is reported as a negative number. Bolded numbers indicate the high score between the two algorithms. The last row reports the averages of each column.

The difference between the high and the low test set scores is striking, over 12 percent points on average, especially when one considers that the high and low training set scores of the 100 SLPs generally differ by at most 1 or 2 percentage points. Wagging outperforms bagging on 8 of the data sets, while bagging betters wagging 3 times. But only 2 of the scores have an associated confidence level which is better than 90 percent (sonar and bcw). While it can be said with greater than 95 percent confidence that both bagging and wagging are better than the average scores of the 100 SLPs, overall it cannot be said with confidence that wagging is better than bagging (or visa versa) on these data sets. However, it is better to use wagging since it reduces the amount of storage and the amount of computation. It is also easier for a human to analyze and understand a system composed of a single SLP, rather than one composed of a hundred SLPs.

3.4 Wagging vs other learning algorithms

Wagging does surprisingly well in comparison with other well-know machine learning and neural network algorithms. Table 4 compares the average test set accuracy for wagging with the results of several machine learning algorithms on the data sets tested in this paper.

algorithm	average	confidence
wag	87.74	NA
bp	86.90	0.928
per	84.17	0.995
c4	84.94	0.995
c4.5	84.74	0.989
ib1	84.99	0.995
id3	83.39	0.995
mml	85.89	0.879
smml	83.69	0.995
cn2	81.89	0.995

Table 4. wagging vs other methods.

The last column gives the statistical confidence (calculated using the Wilcoxon statistical test) that wagging is better than the other learning algorithms on the data sets tested in this paper. Wagging's average generalization results are higher than any of the other machine learning algorithms on the data sets tested in this paper. While having a high average generalization accuracy across several data sets is desirable, one could argue that it is more desirable for a learning algorithm to be able to outperform several other algorithms on at least a few data sets. The wagged SLP has both of these desirable characteristics, since in addition to having good average performance, it also scores higher than any of the other algorithms on 5 of the 15 data sets. The only other algorithm to have 5 high scores is mml. This shows that a wagged SLP has both the desirable property of high average generalization accuracy, and the ability to produce excellent results on specific data sets.

4. The Delta Rule Training Procedure

A few desirable traits for an iterative network training procedure are:

1. More training should generally lead to better performance on the training set.
2. Performance of the network should become more stable as training progresses.
3. There should be a strong positive correlation between training and test set performance, particularly as training set scores approach their maximum.

In this section the delta rule training procedure [12] for an SLP is tested on real world problems to see how well it conforms to the above goals for a network training procedure. Empirically, it is shown that the BWV is better than the most recent weight vector in terms of generalization performance. It is also shown that the BWV, while better than the most recent weight vector, does not in general exhibit the best generalization performance of all the weight vectors produced during the training procedure.

The usual procedure for training an SLP is to apply the delta rule training algorithm until either the network has converged to a solution or a maximum number of iterations has been reached. We say that a network has converged to a solution when there is no longer any error on the training set, or when the total sum squared error (TSSE) on the training set has dropped below a user defined threshold, where the total sum square error is defined as

$$\sum_k (t_k - o_k)^2 \qquad (7)$$

where o_k is defined to be the output of the network for example k. The threshold is generally set to zero for problems with a binary output. Whether the training algorithm halts due to reaching the maximum number of iterations or because the error has dropped below the threshold, the most recent weight vector is generally used by the network for classification of novel examples. There are some inherent problems with this approach, as we will show with the results later in this section. Generally, the training instances are randomly permuted at each iteration of the delta rule algorithm.

Let w_i = weight vector produced after training iteration i

$P(w_i, \mathbf{data})$ = accuracy of w_i on a set of data

Due to the randomness of the training procedure, and also due to the cyclical nature of the delta rule algorithm, it is often observed that

$$\text{for } i > j,\ P(w_i, \text{train}) < P(w_j, \text{train})$$

In other words, there is no guarantee that the current weight vector produced by the delta rule algorithm is better (more accuracte on the training set) than some previously produced weight vector. The hope is that $P(w_i, \text{train})$ will be nearly as good as $P(w_j, \text{train})$, or that as training progresses their will be a high probability that $P(w_i, \text{train})$ will be as good as (or nearly as good as) $P(w_j, \text{train})$. But for most of the real world problems tested in this paper this type of asymptotic, stable performance is not observed. A more important problem is the degree of correlation between $P(w_i, \text{train})$ and $P(w_i, \text{test})$. While this correlation is nearly always positive when taken across all training iterations, the correlation can be negative if we restrict the calculation to, say, the set of w_i which corresponds to the top n training set scores. This means that choosing the single most accurate weight vector will often not maximize test set performance.

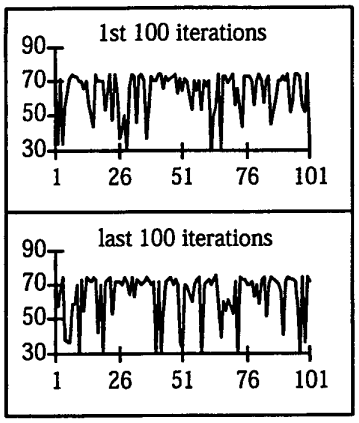

Figure 1. bc training set accuracy.

All of the results in this section were obtained using the standard delta rule training algorithm. The learning rate was set to 1, network weights were updated after the presentation of each pattern, and patterns were randomly permuted for each iteration. Training was halted at 100,000 iterations. This first item we examine is the variability in training accuracy as the training procedure progresses. Figure 1 shows the training set accuracy of an SLP for both the first and last 100 epochs of a single training run on the breast-cancer data set. This training run is typical of what occurs when training an SLP on the breast-cancer data set. The most striking thing about these two training segments is the lack of any significant visual feature which could be used to distinguish between them, despite the fact that there are over 99,000 training epochs separating them. Each training segment contains several sharp downward spikes to around 30% accuracy, separated by plateaus where the accuracy hovers at just above 70%.

The best training set score achieved during this training run is 77.43, which occurred once at the 89,878th epoch. On average, stopping at some arbitrary point during the last 100 epochs will produce a weight vector which is over 12 percentage points less accurate on the training set than the best weight vector generated during the training run. The variability of training set scores seen with the bc data set is typical of the real world data sets tested in this paper. Assuming that these results hold for general real world problems, this means that using the most recent weight vector (after stopping at a maximum iteration count) will generally lead to a network with subpar performance in terms of training set accuracy.

It is relatively simple to solve the problem of variable end-of-epoch training set scores by testing the network at the end of each epoch and saving the weight settings which produce the highest accuracy on the training set. The penalty that is incurred by this procedure is an approximate two-fold increase in computational complexity. The best weight vector is saved on the assumption that high training accuracy correlates well with high test set accuracy. While this assumption does not always hold, it is still usually better to use the weight vector which produced the highest training set accuracy during the training phase than it is to use the most recent weight vector produced at the end of the training phase.

Figure 2 shows the average performance (top 20 training scores and associated average test set scores) for all of the data sets. The test accuracy peaks (on average) at the 4th highest training set score, and thereafter decreases. So most of the data sets exhibit a negative correlation if the calculation is restricted to use only the top 3 or 4 training set scores and associated test set scores. Of the 15 data sets, only 4 of the sets had an average test set accuracy which peaked at the highest training set score.

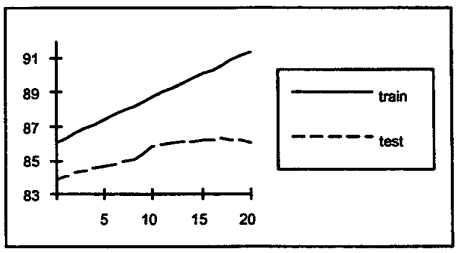

Figure 2. Top 20 training set vs test set scores.

Even though the weight vector which has the highest training set score tends to lead to overfitting, it still produces better test results than using the most recent weight vector. The average test set accuracy of the most recent weight vector (the one produced at the end of the 100,000th epoch) across all of the data sets is 83.91 percent, while BWV has a test set accuracy of 86.61 percent. BWV is in turn outperformed by wagging by 1.2 percentage points on average.

5. Discussion

It is surprising that an SLP actually outperforms all of the other "more capable" learning algorithms. There are a number of possible explanations for this result. There is some indication that it is difficult for a single learning algorithm to outperform all other learning algorithms on a large variety of problems [15]. If the only information that is given is the data, then it is possible to prove that no machine learning algorithm will outperform any other algorithm. So, the more problems that one tests on, the more likely it is that a more complex learning algorithm will have a difficult time beating the average performance of a simple learning algorithm, or visa versa. The fact that the SLP outperforms the other algorithms may only mean that the data sets tested in this paper are well suited to the SLP, and that there are other real world problems which the SLP will perform poorly on.

Whether or not one algorithm will be able to perform better than all others on real world problems depends on what types of characteristics real world problems have, their tendencies and so forth. It is possible that the better performance of an SLP on the problems tested in this paper is due to the data sets having few high order correlations between inputs and output classification. Evidence of this lack of higher order correlations in real world data sets can be seen in the relatively good

performance of the extremely simple 1-rules learning algorithm ([6]), and in the performance of the wagged SLP. More often than not, the features that people choose when designing a real world data set will be those which exhibit first order correlations with the output classification, and potentially little if any higher order correlation's. Learning algorithms that look for, or are capable of handling higher order correlations will tend not to perform as well on data sets which have no such correlations, since any higher order correlations that they 'find' will not be valid.

Even when there are higher order correlations in the data this does not mean that an algorithm which can find such correlations will do any better than one which cannot. The problem is that the learning algorithm must find the *right* higher order correlations, or nearly the right ones, and not just ones that match the training data in order for them to be of benefit. The learning algorithm that one chooses will bias the type of higher order correlations which are most likely to be discovered, and the correlations it tends to discover may or may not be appropriate for particular problem domains.

6 Conclusion

Using the BWV significantly improves generalization performance over using the most recent weight vector on the data sets tested in this paper. Further significant improvement in generalization accuracy can be made by wagging or averaging BWVs from several different runs on the same training data. The weights in the BWV which are responsible for poor performance tend to have high variance between separate training runs on the same training data. Wagging can be viewed as an attempt to average out these weights so that their effect will be minimized. The improvement in test set accuracy that wagging makes on the data sets used in this paper is slightly better than that gained from bagging, with the advantage over bagging that only a single weight vector need be stored. In addition, wagging produced test set scores which are on average better than any of the other learning algorithms compared in this paper.

It should be possible to further improve the generalization results for an SLP. We did not implement any procedure (other than limiting the maximum number of training iterations) to keep weight magnitudes relatively equal across separate training runs. When averaging weights, if one of the SLPs has extremely large weights in comparison with the other SLPs then it will tend to dominate the average. By not implementing a procedure to keep weight magnitudes more uniform, this could lead to a situation where the average weights perform no differently than the SLP that has the largest weights. So, it may be possible to improve the generalization results of the wagged SLP even further by normalizing the weight vectors before averaging them, or perhaps by using some form of weight decay to keep weights from having overly large magnitude.

Another area where improvements can be made is in the training procedure itself. A more stable, asymptotic training method might lead to better weight vector estimates and thus a better average weight vector. We have done some experiments where we have tried to smooth the transition from one weight vector to the next during training by using a local windowed averaging scheme. By combining this smoothing technique with a momentum term we have been able to improve convergence times by an order of magnitude on the sonar data set (from over 40,000 iterations down to about 2,000).

Bibliography

[1] Aha, D W, D Kibler and M K Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6, 37-66.

[2] Breiman, L. 1996. Bagging Predictors. *Machine Learning* 24, 123-140.

[3] Buntine, W. 1989. *Learning classification rules using bayes. Proceedings of the sixth international workshop on machine learning* Morgan Kaufman Publishers, San Mateo CA 94-98.

[4] Clark P. and T. Niblett. The CN2 Induction Algorithm. Machine Learning, 3(4):261-283, 1989.

[5] Friedman, J, T Hastie, and R Tibshirani. 1998. Additive logistic regression: a statistical view of boosting. Tech. report July 23, 1998

[6] Holte, R C. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, vol. 3, pp. 63-91.

[7] Kearns, M, Y Mansour, A Y Ng, and D Ron. 1995. An Experimental and Theoretical Comparison of Model Selection Methods. *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pp 21-30.

[8] Maclin, R and D Opitz. 1997. An empirical evaluation of bagging and boosting. *The Fourteenth National Conference on Artificial Intelligence.*

[9] McClelland, J L and D E Rumelhart. 1988. *Explorations in Parallel Distributed Processing*. The MIT Press, Cambridge Massachusetts.

[10] Quinlan, J R. 1993. *C4.5: Programs for Machine Learning*. Pat Langley editor. Morgan Kaufman publisher.

[11] Rissanen, J. 1986. Stochastic complexity and modeling. *The Annals of Statistics* 14, no 3, 1080-1100.

[12] Rosenblatt, F. 1960. Perceptual generalization over transformation groups. *Self Organizing Systems*. Pergamon Press, New York 63-96.

[13] Schaffer, C. 1993. Overfitting avoidance as bias. *Machine Learning*, 10, 153-178.

[14] Utans, J. . Weight averaging for neural networks and local resampling schemes.

[15] Wolpert, D H. 1993. On Overfitting Avoidance as Bias. Technical Report SFI TR 92-03-5001. Santa Fe, NM: The Santa Fe Institute.

[16] Zarndt, F. 1995. A Comprehensive Case Study: *An Examination of Machine Learning and Connectionist Algorithms*. Masters Thesis Brigham Young University.