Overfitting and Generalization in Learning Discrete Patterns

Charles X. Ling
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
E-mail: ling@csd.uwo.ca
(519)-661-3341 (o), (519)-661-3515 (fax)

Abstract

Understanding and preventing overfitting is a very important issue in artificial neural network design, implementation, and application. Weigend (1994) reports that the presence and absence of overfitting in neural networks depends on how the testing error is measured, and that there is no overfitting in terms of the classification error (symbolic-level errors). In this paper, we show that, in terms of the classification error, overfitting does occur for certain representation used to encode the discrete attributes. We design simple Boolean functions with clear rationale, and present experimental results to support our claims. In addition, we report some interesting results on the best generalization ability of networks in terms of their sizes.

To appear as Neurocomputing Letter in *Neurocomputing: An International Journal.* A long version of this paper can be ftp'd at ftp.csd.uwo.ca at pub/ling/papers/tr-overfitting-long.ps.Z.

1 Introduction

The problem of overfitting has drawn significant attention recently (cf. (Weigend, 1994; Wang, Venkatesh, & Judd, 1994; Grossman, 1994)). The operational definition of overfitting is that the error on the testing sample increases if training goes on for too long. Therefore, training should be stopped at an appropriate point, rather than allowing to proceed until the training error is as small as possible (e.g. near-zero). Contradictory results, however, are reported even about the mere presence and absence of the overfitting effect. Weigend (1994) discovered that in a text-to-phoneme mapping learning, whether overfitting occurs depends on how the testing error is measured. He shows that with the cross-entropy error[1] overfitting is very evident, but that no overfitting is observed with the sum squared error and the sum absolute error. Since his problem concerns with a symbolic mapping, the error that defines overfitting should be at symbolic level; however, the three error measurements he uses are based on the numerical difference between the network outputs and the target output patterns.

We study the general problem of overfitting in learning symbolic patterns in this paper. Many applications of neural networks concern symbolic pattern learning (such as text-to-speech mapping (Sejnowski & Rosenberg, 1987), or verb past-tense learning (Rumelhart & McClelland, 1986; MacWhinney & Leinbach, 1991; Ling & Marinov, 1993)). Since we are learning discrete (symbolic) patterns, the error we really *care about* is the *symbolic-level error*, or the *classification error* of discrete patterns. Therefore, unless otherwise indicated, we use the term *error* in the rest of the paper as the classification error, and *overfitting* as an increase in the classification error on the testing sample.

2 Overfitting and Generalization

To study the overfitting of learning discrete mappings, a small Boolean function of 7 variables is created and used in this section. The design is very simple but with clear rationale. The results appear to be crystal-clear, and can be verified easily.

[1]The error for each pattern is chosen to be the logarithm of the activation value of the output unit that corresponds to the target class (Weigend, 1994).

To create the target function to be learned, the following 7-variable Boolean function f is first chosen:

$$c = 2v_1 - v_2 + 3v_3 + 2v_4 - v_5 + 3v_6 + v_7,$$
$$\text{if } (2 < c < 7) \text{ then } f = 0; \text{ else } f = 1.$$

The actual target function is defined by the Boolean function f except 10% of function values chosen randomly are flipped (0 to 1 and 1 to 0). That is, the "noise" is inherited in and is *regarded* as part of the target function. The rationale is to create *both* a primary regularity (i.e., easily captured by networks) based on the linear relation c and a secondary regularity. The secondary "regularity" can be any ill-posed, irregular problem for the networks, such as those flipped values. Most domains contain both primary and second regularities. For example, there are some irregular past tenses among mostly regular past tenses in learning the past tense of English verbs; or exceptional mappings among regular ones in text-to-speech conversion. To create training and testing samples for experiments, the 128 possible examples are split randomly into a training set and a testing set of equal sizes without overlap. It happens that the training set contains 4 of the flipped examples, while the testing set contains 7. Therefore, the minimum predictive error on the testing sample one can expect is 7 out of 64.

2.1 Representation Issues

In artificial neural networks, a symbolic pattern is normally represented as a string of binary bits. Two representation schema are possible for symbolic attributes. One obvious way to represent this Boolean function is a 7-to-1 mapping: 7 Boolean input attributes to 1 Boolean output attribute. This is a special (extreme) case of the distributed representation[2] (Hinton, McClelland, & Rumelhart, 1986). However, from another perspective, this function can be regarded as realizing a mapping from *one* input attribute that takes 128 values to one output attribute that takes 0 and 1. In this case, it is a 1-to-1 mapping. This is sometimes called *local representation*, and we call it *one-per-value* representation in this paper. For one-per-value encoding, a string of 128 bits is used for the input. For any input attribute value, only one bit that corresponds to that value is 1; all the rest of the 127 bits are 0.

[2]Normally, more than 7 bits are used to encode 128 values.

2.2 Representations that Exhibit Overfitting

Networks with 7 input units and 1 output unit are used for the 7-to-1 mapping representation. A neural network package, called Xerion, developed at University of Toronto is employed in training. The standard error backpropagation is used here with a small learning rate (0.1) and no momentum term, to avoid artifacts of the learning curve (similar to (Weigend, 1994)). To observe the overfitting effect, errors on the testing examples[3] are monitored every 10 epochs until 50,000 epochs. The initial weights are set randomly (uniformly distributed) within $[-0.3, 0.3]$. We tested networks with 1 (too small), 2, 3, 4 (adequate), 6, 8, 12, 20, 64, 100 (overly large) hidden units. All results are averaged over 10 runs (unless otherwise indicated in the table) with different random connection weight seeds for network initialization. Table 1 summarises the average testing errors and the overfitting phenomena for the runs that converge[4].

Results in Table 1 are obtained with initial weights set within $[-0.3, 0.3]$. However, it is a common practice to use very small initial weights in training large networks (Kolen & Pollack, 1990). In addition, it is also a common practice to use the weight decay technique and the conjugate gradient search method. All of these experiments are run and the results are very similar to the one presented in Table 1 (for details, see `tr-overfitting-long.ps.Z` under `pub/ling/papers` on `ftp.csd.uwo.ca`). Combining these results, we can draw the following conclusions about overfitting:

- As also reported by Weigend (1994), overly small networks (too few hidden units to train the network to near-zero error), also overfit, although with very small amount. For example, the testing error for the Boolean function in a typical run goes to a minimum of 30, and raises to 33.

[3]Note that, strictly speaking, the training set should be split into two disjoint subsets, one for training, one for cross-validation. The best network is then tested on the testing examples. In our experiments (also as in (Weigend, 1994)), we monitor the trained network on the testing examples directly. The testing set can be regarded as the validation set.

[4]Training is regarded as converged if the number of training errors is less than or equal to 4 at 50,000 epochs. This number is chosen since there are 4 flipped values in the training set, as indicated earlier in this section. If the network is too small, or if the search is trapped in local minima, the number of training errors would be larger than 4, and such networks would have large testing errors. The results of such cases are not included in the calculation of the average number of testing errors presented in the tables.

- Unlike Weigend's finding, overfitting in terms of the classification error is evident using this representation — it occurs in *every run* we conducted.

We also obtain some very interesting and surprising results on the generalization power of networks of different sizes.

- Networks that are just large enough (which has just enough hidden units to train to near-zero error; 2 and 3 in this case) have small minimum testing errors, and so do large networks of *appropriate* sizes (such as greater than 12). In addition, such networks (just large enough and very large) have more stable generalization (i.e., small standard deviations). Stable generalization is important because training large networks takes a lot of time, and sometimes we cannot afford to make multiple runs for choosing the best network.

- Although it has been reported recently that large networks generalize well (Weigend, 1994; Grossman, Meir, & Domany, 1988), we find that not all large networks do. Not only do networks with medium sizes (between 4 and 12) have larger average minimum testing errors, but their values vary more widely with different random network initialization. Therefore, such networks should be avoided.

- Very large networks (such as with 100 hidden units) sometimes fail to converge, and they take a much longer training time.

2.3 Representations that Exhibit No Overfitting

It is well known that a good representation makes learning easy. Instead of finding a representation that makes the learning of the 7-variable Boolean function easier, we simply modify the target function so that it has only one simple primary regularity. We chose a simple 7-variable function g as $g = (x_1 + x_2 + ... + x_7 > 3)$ and there is no flipped function values (noise). Since the target function is now a purely linear function without "noise", there is no "secondary regularity" in the data. Therefore, we should observe no overfitting. Experiments reported in Section 2.1 were repeated on g and indeed, no overfitting was observed. We also reran these experiments with a few function values of g flipped, introducing a secondary regularity (noise) as part of the target function. Overfitting is observed again.

Another form of representation that does not capture a primary and a secondary regularities, and in fact, does not capture any regularity at all, is the one-per-value representation. We only use this trivial representation to demonstrate our point of view. With 128 input attribute values in the one-per-value encoding, the linear "regularity" existing in the original 7-variable Boolean function disappears! More importantly, the secondary regularity — those with the flipped function values — becomes just as "normal" as others. In fact, networks with no hidden units (i.e., Perceptrons) are sufficient to represent *any* set of training examples: If the network is initiated with very small random weights, only those weights that connect to the input units turned on by positive training examples are increased during training. Such networks, after nominal training, would simply become a "memory" device that remembers only positive examples (output 1) in the training set; everything else is predicted as negative (0). Therefore, the testing accuracy, after the network is nominally trained, remains large and constant, no matter how much more training is applied. Therefore, there will be no overfitting at all. This was experimentally verified.

2.4 A Brief Analysis of Overfitting

We have demonstrated experimentally that the occurrence of overfitting depends on whether the representation allows for the capture of primary and secondary regularities in the data. A sensible explanation for overfitting is that, if the representation is allowed to capture a primary regularity and a secondary regularity, then a network with a proper size and proper training would, at a certain point, capture most of the primary regularity before picking up the secondary one. This network would have the best predictive accuracy for the primary regularity in the testing examples. Excessive training causes the network to fit the secondary regularity as well, which in turn interferes with and *worsens* the prediction of the primary regularity in the testing examples. The result of these is an increase in the testing errors — the overfitting phenomenon. Our analyses have been clearly verified in the Boolean functions with and without the overfitting effect in the previous sections.

With the distributed representation but no flipped values (e.g., g), or with the one-per-value representation, there is no secondary regularity in the target function. Therefore, there is no overfitting problem. The disadvantage of one-per-value representation is that it does not capture any regularity, and thus results in very high testing errors. A good distributed representation uses a small number of bits to capture micro-features among attribute values for the primary regularity, and thus has better generalization accuracy (Hinton et al., 1986). The trade-off introduced is that it may suffer from overfitting since these micro-features may inevitably cast unwanted secondary regularity (noise). A good strategy of avoiding overfitting is to stop training appropriately. The common practice is to use the cross-validation technique — monitoring the predictive error on an independent validation set of examples, and stop training when the minimal validation error is reached.

3 Conclusions

We design simple experiments on Boolean functions with clear rationale and show that the degree of overfitting at symbolic level depends on the representation. If the representation allows the networks to capture *both* a primary regularity and one or more secondary regularities, overfitting occurs after learning the primary regularity. The fitting of the secondary regularities interferes with and thus worsens the prediction of the primary regularity in the testing set. We also observed in series of experiments that no matter what training algorithms are used, the networks that are just large enough (just enough hidden units for near-zero training error), or overly large (e.g., roughly four times larger than just enough) have better and more stable generalization than networks of the size in between.

Acknowledgements

I am grateful to Tom Dietterich, Dave Touretzky, Andreas Weigend, and Billy Schmidt for insightful discussions, and to many people (such as Tal Grossman, Wray Buntine, William Finnoff, Fabien Moutarde, John Kolen, Barak Pearlmutter) who answered and discussed my questions posted in the Connectionist List. I also thank reviewers for their insightful comments on the paper. Danny Silver and Drew van Camp helped in setting up Xerion.

Reference

Grossman, T., Meir, R., & Domany, E. (1988). Learning by choice of internal representations. *Complex Systems*, *2*, 555–575.

Grossman, T. (1994). Use of bad training data for better predictions. In Cowan, J. D., Tesauro, G., & Alspector, J. (Eds.), *Advances in Neural Information Processing Systems — 6 (NIPS*93).*

Hinton, G., McClelland, J., & Rumelhart, D. (1986). Distributed representation. In Rumelhart, D., McClelland, J., & the PDP Research Group (Eds.), *Parallel Distributed Processing Vol 1*, pp. 77 – 109. Cambridge, MA: MIT Press.

Kolen, J. F., & Pollack, J. B. (1990). Backpropagation is sensitive to initial conditions. *Complex Systems*, *4*(3), 269–280.

Ling, C. X., & Marinov, M. (1993). Answering the connectionist challenge: a symbolic model of learning the past tense of English verbs. *Cognition*, *49*(3), 235–290. Anonymous ftp: /pub/ling/papers/cognition-verb.ps.Z on ftp.csd.uwo.ca.

MacWhinney, B., & Leinbach, J. (1991). Implementations are not conceptualizations: Revising the verb model. *Cognition*, *40*, 121 – 157.

Rumelhart, D., & McClelland, J. (1986). On learning the past tenses of English verbs. In Rumelhart, D., McClelland, J., & the PDP Research Group (Eds.), *Parallel Distributed Processing, Vol. 2*, pp. 216 – 271. Cambridge, MA: MIT Press.

Sejnowski, T., & Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, *1*, 145 – 168.

Wang, C., Venkatesh, S. S., & Judd, J. S. (1994). Optimal stopping and effective machine complexity in learning. In Cowan, J. D., Tesauro, G., & Alspector, J. (Eds.), *Advances in Neural Information Processing Systems — 6 (NIPS*93).*

Weigend, A. S. (1994). On overfitting and the effective number of hidden units. In Mozer, M. C., Smolensky, P., Touretzky, D., Elman, J. L., & Weigend, A. S. (Eds.), *Proceedings of the 1993 Connectionist Models Summer School*, pp. 335 – 342. Erbaum Associates.

Table 1: Summary of the overfitting in learning discrete patterns. Standard BP, with initial weights $[-0.3, 0.3]$ and learning rate 0.1.

# of h.u.	percent converged	range of min T.E.	average min T.E.	overfit at 50000 epochs	percent of overfitting
1	0	29 – 30	N/A	N/A	N/A*
2	10	7 – 11	9.0 ± 0.8	3.0 ± 0.8	33.3^{+}
3	50	8 – 11	9.2 ± 1.6	3.4 ± 1.3	37.0
4	100	11 – 16	13.4 ± 2.1	3.6 ± 4.1	26.9
6	100	12 – 17	13.5 ± 1.9	5.0 ± 4.1	37.0
8	100	11 – 17	12.3 ± 2.1	5.3 ± 2.5	43.1
12	100	8 – 13	10.2 ± 1.7	9.3 ± 4.7	91.1
20	100	8 – 12	9.7 ± 1.4	7.8 ± 4.5	80.4
64	100	9 – 11	10.1 ± 0.9	4.9 ± 2.3	48.5
100	50	9 – 13	10.6 ± 1.5	4.8 ± 3.2	45.3

* Even though the network has never converged, a small amount of overfitting (about 10%) is observed.

+ Based on 40 runs with different random seeds.

The first column is the number of hidden units (# of h.u.). The second column indicates the percent of runs that converge. Column 3 records the range of the numbers of minimum testing errors (min T.E.) during training for runs that converge, and Column 4 is the average (and the standard error) of such numbers. (Note that these numbers are not percents of errors, instead, they are just numbers of misclassified testing examples). Column 5 is the amount of overfitting at 50,000 epochs; i.e., the average number of the *increase* in the testing errors from the minimum testing errors. (This represents how much the curve for the testing error "tips up".) The last column is the percent of overfitting, which is simply the ratio of mean values in column 5 and column 4.