The Epoch Interpretation of Learning

John G. Carney and Pádraig Cunningham
Department of Computer Science
University of Dublin
Trinity College
Ireland
John.Carney,Padraig.Cunningham@cs.tcd.ie

Abstract

In this paper we propose a simple, alternative interpretation of back-propagation learning. We call this the "epoch interpretation of learning" and show how it can be used to improve the performance of early-stopping based techniques used for improving generalization performance in neural networks. Experiments performed on noisy, non-linear foreign exchange rate data demonstrate that networks built using an early-stopping technique that uses the epoch interpretation of learning on average out-perform networks built using a conventional early-stopping technique by 11%.

1 Introduction

The ultimate goal of the neural network researcher is to build networks that provide optimal generalization performance. Once a network is trained, we wish it to perform well on examples that are not included in the training set. There are a large variety of techniques described in the literature that attempt to do this. One of the most popular is *early-stopping*. The basic idea of early-stopping is to terminate training when some estimate of generalization error begins to increase. If training is not stopped early and continues to convergence then there is a danger that the network will over-fit its training data and will not generalizise well to new, unseen examples.

In this paper we build upon the principles of early-stopping to develop an alternative interpretation of learning. We show how this "epoch interpretation of learning" can be used to develop techniques that improve generalization performance more effectively than conventional early-stopping approaches. Experiments performed on highly non-linear, noisy foreign exchange rate data provide empirical evidence that using the epoch interpretation of learning can significantly improve the generalization performance of neural networks.

2 The Epoch Interpretation of Learning

The epoch in back-propagation learning is one weight update or training iteration (see figure 2 overleaf). For each epoch, the back-propagation learning algorithm builds a different model (i.e.) a network with a different set of weights. If a neural network is trained to 1000 epochs, the learning algorithm investigates or moves through 1000 different models. Neural network learning then, can be viewed as a search through a large number of models, for a model that has the set of weights that will provide the best generalization performance. The number of hidden units a network has also influences the generalization performance of a neural network and can be easily incorporated into this view of learning.

To provide a more rigorous expression of these ideas, let us assume we have a training set T and that we use this training set in conjunction with the back-propagation learning algorithm to estimate the weights $\hat{\omega}$ of a network that has a fixed number of hidden units. Let us denote this network as:

$$\phi(T;\hat{\omega}) \tag{1}$$

Using the terminology of [3], a specific $\hat{\omega}$ can be thought of as a point in an abstract space of all possible $\hat{\omega}$'s called *weight-space*. During learning a large number of $\hat{\omega}$'s are investigated by the learning algorithm. The $\hat{\omega}$'s that we choose should be those that provide the best generalization performance. To do this however we must first "index" each position the learning algorithm follows through weight-space during training[1]. We need to do this because, once a network's training is converged and overfitting is observed, we will need to re-train it, terminating training at the point where the $\hat{\omega}$'s correspond to the position in weight-space that provides estimated optimal generalization performance for a training session[2]. We propose that the epoch should be used to index weight-space in this way. We represent then, the evolution of neural network models during back-propagation learning as the set:

$$\Upsilon = \{\phi(T;\hat{\omega}_e)\}_{e=1}^{E} \tag{2}$$

where e is the current epoch and E is the maximum number of epochs specified by the user[3]. Using the epoch in this way is powerful because it allows us to connect the abstract concept of a neural network's journey through weight-space during training to something we can use and relate to in practice. It takes advantage of the fact that there is a direct mapping between the epoch and neural model parameters and their values.

Equation (2) above, however, ignores the number of hidden units a network has. To incorporate the number of hidden units into this framework, we extend equation (2) to a set of sets

$$\Upsilon = \{\{\phi(T;\hat{\omega})\}_{e=1}^{E}\}_{h=1}^{H} \tag{3}$$

where H is the maximum number of hidden units specified by the user[4]. For each h in equation (3), the algorithm follows a path through a different weight-space. The goal of the modeler should be to find the values for e and h that provide the best generalization performance. In the next section we show how a variation of cross-validation [4] can be used to do this.

[1] In theory, there are an infinite number of paths through weight-space. We only attempt to index a *single* path (i.e.) for a specific training session. The route this path follows through weight-space and position from which it starts depends on the initial values of the $\hat{\omega}$'s, the training set T and other user defined parameters such as learning and momentum rate.

[2] A more efficient solution would be to save in memory or on disk each "best" set of weights ((i.e.) the weights that correspond to minimums in generalization error) during training.

[3] A large enough value of E should be chosen to ensure training error converges and overfitting is observed.

[4] A large enough value of H should be chosen to ensure all dynamics of the system under study can be modeled by the network.

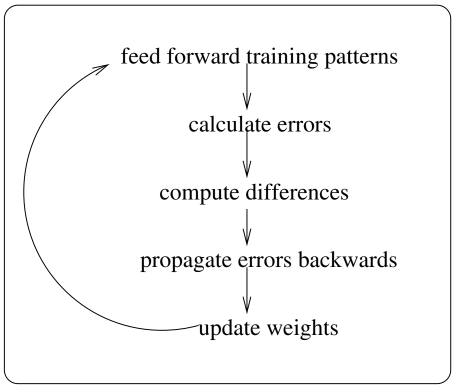

Figure 1: The back-propagation neural network epoch.

3 Optimizing Generalization Performance

In this section we describe and compare two variations of early-stopping. The first is a simple "straw man" version of early-stopping (i.e.) training is terminated as soon as an estimate for generalization error begins to increase for the first time. The second utilizes the epoch interpretation of learning (i.e.) an estimate of generalization error is calculated for every combination of values of e and h. A network is then re-trained, using the combination of values of e and h that provide the best estimated generalization performance, to determine when training should be stopped and how many hidden units should be used. In both cases, the simplest variety of cross-validation, which we call "hold-out validation", is used to estimate generalization error.

Hold-out validation simply consists of using $N - N_c$ of the available training samples assembled in T_{train} for estimating the network weights $\hat{\omega}_{N-N_c}$, and the remaining N_c samples assembled in T_{cross} for calculating the generalization error estimate. This hold-out validation generalization error estimate can be expressed in terms of the neural network cost function as:

$$G_{hov} = \frac{1}{N_c} \sum_{n=N-N_c+1}^{N} (y_n - \phi(x_n; T_{N-N_c}; \hat{\omega}_{N-N_c}))^2 \tag{4}$$

which can be more elegantly expressed as

$$G_{hov} = \frac{1}{N_c} \sum_{n=N-N_c+1}^{N} E(y_n; x_n; \hat{\omega}_{N-N_c}) \tag{5}$$

For conventional early-stopping, G_{hov} is measured at regular intervals during training. When it begins to increase, training is terminated.

This estimate of when training should be stopped, however, will not work when generalization fluctuates as illustrated in figure 1. Also, no provision is made for how many hidden units should be used. To address these issues we use the epoch interpretation of learning to extend equation (5) to the following:

$$G_{hov}(e, h) = \frac{1}{N_c} \sum_{n=N-N_c+1}^{N} E(y_n; x_n; \hat{\omega}_{N-N_c}(e, h)) \tag{6}$$

Here, $G_{hov}(e, h)$ is the hold-out validation generalization error estimate of a network with h hidden units, trained to epoch e. Values for $G_{hov}(e, h)$ are found for $e = 1, .., E$ and

	HV	*EIL-HV*	*% Decrease*
chf/jpy	1.125	0.969	14%
gbp/dem	0.048	0.042	13%
usd/dem	0.014	0.012	7%
usd/jpy	1.314	1.212	8%

Table 1: The root mean squared error test set performance of neural networks trained using conventional hold-out validation (HV) compared to the performance of networks trained using a variation of hold-out validation that uses the epoch interpretation of learning (EIL-HV). On average, the percentage decrease in error yielded by the epoch interpretation of learning is 11%.

$h = 1, .., H$. The user then chooses the values for e and h that provide the best estimated generalization performance:

$$OPT(e, h) = \overset{argmin}{h, e} \quad (G_{hov}(e, h)) \tag{7}$$

The network is then re-trained using $OPT(e, h)$ to estimate when training should be stopped and how many hidden units the network should have. Equivalently, the network saved on disk corresponding to these values for e and h is chosen as the optimal network.

4 Experiments

In this section, we present the results of 40 experiments that compare the performance of networks built using conventional early-stopping estimates (equation (5)) to estimates that use the epoch interpretation of learning (equations (6) and (7)).

The 4 data-sets chosen for the experiments described in this section are typical noisy foreign exchange rate data-sets. Each contains 1200 daily cross-rates from 17/6/92 to 12/3/97 and is combined with daily volatility and interest rate market information to create training vectors that encode a simple 5-day lag-space. These are arranged into pools of 1200 training samples for each data-set. Each of these training samples contains a vector of 7 inputs (5 foreign exchange cross-rates, a volatility and an interest rate differential input) and 1 output (cross-rate 5 days ahead).

For each data-set and technique compared, experiments were performed on 5 different random re-samples of training, validation and test set (i.e.) each entry in table 1 above corresponds to an average of errors across 5 experiments. The validation and test sets were formed by sampling at random *without* replacement 600 samples from each data-set pool, using 500 of these for validation, and 100 for testing. The remaining 500 samples were used for training. This random re-sampling of test sets is not normally done in time series prediction experiments – one usually trains on the past and tests on the future. However, given a limited amount of useful time-series data this technique suffices as a method to increase the statistical significance of results.

Results are summarized in table 1, overleaf.

5 Conclusion

In this paper we introduced a simple, alternative interpretation of neural network learning and showed how it can be used to improve the performance of an early-stopping technique based on hold-out validation. We argued that this performance improvement is most noticeable when noisy non-linear training data is used.

The aim of this paper is to provide a simple introduction to the epoch interpretation of learning. It should be clear that the ideas presented here can be extended so that the epoch interpretation of learning can be used to improve the performance of other early-stopping approaches such as those based on *k*-fold cross-validation for example. In [2], we show how the epoch interpretation of learning can be used to improve the performance of an early stopping based technique for optimizing bagged neural network performance.

The epoch interpretation of learning allows one to estimate the best set of weights for a specific training session (i.e.) for a *single* path through weight-space. It will not necessarily find the absolute best set of weights among the complete set in weight-space. However, assuming reasonable parameters such as learning and momentum rate are chosen, and not too many deep local minima exist, the back-propagation algorithm, should come close to these "best" set of weights during its journey through weight-space. Assuming a technique that can accurately estimate generalization error is used, estimating the best set of weights for a specific training session, in the manner in which the epoch interpretaion of learning attempts to do it, will be useful.

References

[1] S. Amari, N. Murata, K.-R. Muller, M. Finke and H. Yang. Statistical theory of over-training - is cross-validation asymptotically effective? In M. Mozer, M. Jordan and T. Peskes, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, 1996. MIT Press.

[2] J. Carney and P. Cunningham. The NeuralBAG algorithm: Optimizing generalization performance in bagged neural networks. University of Dublin, Trinity College technical report, TCD-CS-1998-23.

[3] J. Hertz, A. Krogh and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, California, 1991.

[4] M. Stone. Cross-validatory choice and assesment of statistical predictors. *J. R. Statistical Society*. B36, 111-147.