In *Advances in Neural Information Processing Systems 4*,
J.E. Moody, S.J. Hanson and R.P. Lippmann, eds.
Morgan Kauffmann Publishers, San Mateo CA, 1992, p. 950–957.

# A Simple Weight Decay Can Improve Generalization

**Anders Krogh***
CONNECT, The Niels Bohr Institute
Blegdamsvej 17
DK-2100 Copenhagen, Denmark
krogh@cse.ucsc.edu

**John A. Hertz**
Nordita
Blegdamsvej 17
DK-2100 Copenhagen, Denmark
hertz@nordita.dk

## Abstract

It has been observed in numerical simulations that a weight decay can improve generalization in a feed-forward neural network. This paper explains why. It is proven that a weight decay has two effects in a linear network. First, it suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem. Second, if the size is chosen right, a weight decay can suppress some of the effects of static noise on the targets, which improves generalization quite a lot. It is then shown how to extend these results to networks with hidden layers and non-linear units. Finally the theory is confirmed by some numerical simulations using the data from NetTalk.

## 1 INTRODUCTION

Many recent studies have shown that the generalization ability of a neural network (or any other 'learning machine') depends on a balance between the information in the training examples and the complexity of the network, see for instance [1, 2, 3]. Bad generalization occurs if the information does not match the complexity, *e.g.* if the network is very complex and there is little information in the training set. In this last instance the network will be over-fitting the data, and the opposite situation corresponds to under-fitting.

*Present address: Computer and Information Sciences, Univ. of California Santa Cruz, Santa Cruz, CA 95064.

Often the number of free parameters, *i.e.* the number of weights and thresholds, is used as a measure of the network complexity, and algorithms have been developed, which minimizes the number of weights while still keeping the error on the training examples small [4, 5, 6]. This minimization of the number of free parameters is not always what is needed.

A different way to constrain a network, and thus decrease its complexity, is to limit the growth of the weights through some kind of weight decay. It should prevent the weights from growing too large unless it is really necessary. It can be realized by adding a term to the cost function that penalizes large weights,

$$E(\boldsymbol{w}) = E_0(\boldsymbol{w}) + \frac{1}{2}\lambda \sum_i w_i^2, \tag{1}$$

where $E_0$ is one's favorite error measure (usually the sum of squared errors), and $\lambda$ is a parameter governing how strongly large weights are penalized. $\boldsymbol{w}$ is a vector containing all free parameters of the network, it will be called the weight vector. If gradient descend is used for learning, the last term in the cost function leads to a new term $-\lambda w_i$ in the weight update:

$$\dot{w}_i \propto -\frac{\partial E_0}{\partial w_i} - \lambda w_i. \tag{2}$$

Here it is formulated in continuous time. If the gradient of $E_0$ (the 'force term') were not present this equation would lead to an exponential decay of the weights.

Obviously there are infinitely many possibilities for choosing other forms of the additional term in (1), but here we will concentrate on this simple form.

It has been known for a long time that a weight decay of this form can improve generalization [7], but until now not very widely recognized. The aim of this paper is to analyze this effect both theoretically and experimentally. Weight decay as a special kind of regularization is also discussed in [8, 9].

## 2 FEED-FORWARD NETWORKS

A feed-forward neural network implements a function of the inputs that depends on the weight vector $\boldsymbol{w}$, it is called $f_w$. For simplicity it is assumed that there is only one output unit. When the input is $\boldsymbol{\xi}$ the output is $f_w(\boldsymbol{\xi})$. Note that the input vector is a vector in the $N$-dimensional input space, whereas the weight vector is a vector in the weight space which has a different dimension $W$.

The aim of the learning is not only to learn the examples, but to learn the underlying function that produces the targets for the learning process. First, we assume that this *target function* can actually be implemented by the network. This means there exists a weight vector $\boldsymbol{u}$ such that the target function is equal to $f_u$. The network with parameters $\boldsymbol{u}$ is often called the *teacher*, because from input vectors it can produce the right targets. The sum of squared errors is

$$E_0(\boldsymbol{w}) = \frac{1}{2}\sum_{\mu=1}^{p}[f_u(\boldsymbol{\xi}^\mu) - f_w(\boldsymbol{\xi}^\mu)]^2, \tag{3}$$

where $p$ is the number of training patterns. The learning equation (2) can then be written

$$\dot{w}_i \propto \sum_\mu [f_u(\boldsymbol{\xi}^\mu) - f_w(\boldsymbol{\xi}^\mu)] \frac{\partial f_w(\boldsymbol{\xi})}{\partial w_i} - \lambda w_i. \tag{4}$$

Now the idea is to expand this around the solution $\boldsymbol{u}$, but first the linear case will be analyzed in some detail.

## 3 THE LINEAR PERCEPTRON

The simplest kind of 'network' is the linear perceptron characterized by

$$f_w(\boldsymbol{\xi}) = N^{-1/2} \sum_i w_i \xi_i \tag{5}$$

where the $N^{-1/2}$ is just a convenient normalization factor. Here the dimension of the weight space ($W$) is the same as the dimension of the input space ($N$).

The learning equation then takes the simple form

$$\dot{w}_i \propto \sum_\mu N^{-1} \sum_j [u_j - w_j] \xi_j^\mu \xi_i^\mu - \lambda w_i. \tag{6}$$

Defining

$$v_i \equiv u_i - w_i \tag{7}$$

and

$$A_{ij} = N^{-1} \sum_\mu \xi_i^\mu \xi_j^\mu \tag{8}$$

it becomes

$$\dot{v}_i \propto -\sum_j A_{ij} v_j + \lambda(u_i - v_i). \tag{9}$$

Transforming this equation to the basis where $\mathbf{A}$ is diagonal yields

$$\dot{v}_r \propto -(A_r + \lambda) v_r + \lambda u_r, \tag{10}$$

where $A_r$ are the eigenvalues of $\mathbf{A}$, and a subscript $r$ indicates transformation to this basis. The generalization error is defined as the error averaged over the distribution of input vectors

$$\begin{aligned} F &= \langle [f_u(\boldsymbol{\xi}) - f_w(\boldsymbol{\xi})]^2 \rangle_\xi = \langle N^{-1} (\sum_i v_i \xi_i)^2 \rangle_\xi = N^{-1} \sum_{ij} v_i v_j \langle \xi_i \xi_j \rangle_\xi \\ &= N^{-1} \sum_i v_i^2. \end{aligned} \tag{11}$$

Here it is assumed that $\langle \xi_i \xi_j \rangle_\xi = \delta_{ij}$. The generalization error $F$ is thus proportional to $|\boldsymbol{v}|^2$, which is also quite natural.

The eigenvalues of the covariance matrix $\mathbf{A}$ are non-negative, and its rank can easily be shown to be less than or equal to $p$. It is also easily seen that all eigenvectors belonging to eigenvalues larger than 0 lies in the subspace of weight space spanned by the input patterns $\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^p$. This subspace, called the pattern subspace, will be denoted $V_p$, and the orthogonal subspace is denoted by $V_p^\perp$. When there are sufficiently many examples they span the whole space, and there will be no zero eigenvalues. This can only happen for $p \geq N$.

When $\lambda = 0$ the solution to (10) inside $V_p$ is just a simple exponential decay to $v_r = 0$. Outside the pattern subspace $A_r = 0$, and the corresponding part of $v_r$ will be constant. Any weight vector which has the same projection onto the pattern subspace as $\boldsymbol{u}$ gives a learning error 0. One can think of this as a 'valley' in the error surface given by $\boldsymbol{u} + V_p^\perp$.

The training set contains no information that can help us choose between all these solutions to the learning problem. When learning with a weight decay $\lambda > 0$, the constant part in $V_p^\perp$ will decay to zero asymptotically (as $e^{-\lambda t}$, where $t$ is the time). An infinitesimal weight decay will therefore choose the solution with the smallest norm out of all the solutions in the valley described above. This solution can be shown to be the optimal one on average.

## 4 LEARNING WITH AN UNRELIABLE TEACHER

Random errors made by the teacher can be modeled by adding a random term $\eta$ to the targets:

$$f_u(\boldsymbol{\xi}^\mu) \quad \longrightarrow \quad f_u(\boldsymbol{\xi}^\mu) + \eta^\mu. \tag{12}$$

The variance of $\eta$ is called $\sigma^2$, and it is assumed to have zero mean. Note that these targets are not exactly realizable by the network (for $\alpha > 0$), and therefore this is a simple model for studying learning of an unrealizable function.

With this noise the learning equation (2) becomes

$$\dot{w}_i \propto \sum_\mu (N^{-1} \sum_j v_j \xi_j^\mu + N^{-1/2} \eta^\mu) \xi_i^\mu - \lambda w_i. \tag{13}$$

Transforming it to the basis where $\mathbf{A}$ is diagonal as before,

$$\dot{v}_r \propto -(A_r + \lambda) v_r + \lambda u_r - N^{-1/2} \sum_\mu \eta^\mu \xi_r^\mu. \tag{14}$$

The asymptotic solution to this equation is

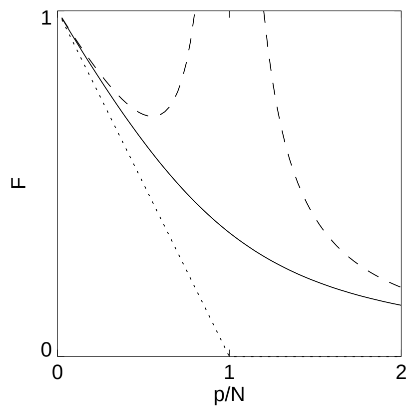
$$v_r = \frac{\lambda u_r - N^{-1/2} \sum_\mu \eta^\mu \xi_r^\mu}{\lambda + A_r}. \tag{15}$$

The contribution to the generalization error is the square of this summed over all $r$. If averaged over the noise (shown by the bar) it becomes for each $r$

$$F_r = \overline{v_r^2} = \frac{\lambda^2 u_r^2 + N^{-1} \sum_\mu (\xi_r^\mu)^2 \overline{(\eta^\mu)^2}}{(\lambda + A_r)^2} = \frac{\lambda^2 u_r^2 + A_r \sigma^2}{(\lambda + A_r)^2}. \tag{16}$$

The last expression has a minimum in $\lambda$, which can be found by putting the derivative with respect to $\lambda$ equal to zero, $\lambda^r_{\text{optimal}} = \sigma^2 / u_r^2$. Remarkably it depends only

Figure 1: Generalization error as a function of $\alpha = p/N$. The full line is for $\lambda = \sigma^2 = 0.2$, and the dashed line for $\lambda = 0$. The dotted line is the generalization error with no noise and $\lambda = 0$.




on $\boldsymbol{u}$ and the variance of the noise, and *not* on $\mathbf{A}$. If it is assumed that $\boldsymbol{u}$ is random (16) can be averaged over $\boldsymbol{u}$. This yields an optimal $\lambda$ independent of $r$,

$$\lambda_{\text{optimal}} = \frac{\sigma^2}{u^2}, \tag{17}$$

where $u^2$ is the average of $N^{-1}|\boldsymbol{u}|^2$.

In this case the weight decay to some extent prevents the network from fitting the noise.

From equation (14) one can see that the noise is projected onto the pattern subspace. Therefore the contribution to the generalization error from $V_p^\perp$ is the same as before, and this contribution is on average minimized by a weight decay of any size.

Equation (17) was derived in [10] in the context of a particular eigenvalue spectrum. Figure fig. 1 shows the dramatic improvement in generalization error when the optimal weight decay is used in this case. The present treatment shows that (17) is independent of the spectrum of $\mathbf{A}$.

We conclude that a weight decay has two positive effects on generalization in a linear network: 1) It suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem. 2) If the size is chosen right, it can suppress some of the effect of static noise on the targets.

## 5 NON-LINEAR NETWORKS

It is not possible to analyze a general non-linear network exactly, as done above for the linear case. By a local linearization, it is however, possible to draw some interesting conclusions from the results in the previous section.

Assume the function is realizable, $f = f_u$. Then learning corresponds to solving the $p$ equations

$$f_w(\boldsymbol{\xi}^\mu) = f_u(\boldsymbol{\xi}^\mu) \tag{18}$$

in $W$ variables, where $W$ is the number of weights. For $p < W$ these equations define a manifold in weight space of dimension at least $W - p$. Any point $\tilde{\boldsymbol{w}}$ on this manifold gives a learning error of zero, and therefore (4) can be expanded around $\tilde{\boldsymbol{w}}$. Putting $\boldsymbol{v} = \tilde{\boldsymbol{w}} - \boldsymbol{w}$, expanding $f_w$ in $\boldsymbol{v}$, and using it in (4) yields

$$\begin{aligned} \dot{v}_i &\propto -\sum_{\mu,j} \left( \frac{\partial f_w(\boldsymbol{\xi}^\mu)}{\partial w_j} \right) v_j \frac{\partial f_w(\boldsymbol{\xi}^\mu)}{\partial w_i} + \lambda(\tilde{w}_i - v_i) \\ &= -\sum_j \mathcal{A}_{ij}(\tilde{\boldsymbol{w}}) v_j - \lambda v_i + \lambda \tilde{w}_i \end{aligned} \tag{19}$$

(The derivatives in this equation should be taken at $\tilde{\boldsymbol{w}}$.)

The analogue of $\mathbf{A}$ is defined as

$$\mathcal{A}_{ij}(\tilde{\boldsymbol{w}}) \equiv \sum_\mu \frac{\partial f_w(\boldsymbol{\xi}^\mu)}{\partial w_i} \frac{\partial f_w(\boldsymbol{\xi}^\mu)}{\partial w_j}. \tag{20}$$

Since it is of outer product form (like $\mathbf{A}$) its rank $R(\tilde{\boldsymbol{w}}) \le \min\{p, W\}$. Thus when $p < W$, $\boldsymbol{\mathcal{A}}$ is *never* of full rank. The rank of $\boldsymbol{\mathcal{A}}$ is of course equal to W minus the dimension of the manifold mentioned above.

From these simple observations one can argue that good generalization should *not* be expected for $p < W$. This is in accordance with other results (cf. [3]), and with current 'folk-lore'. The difference from the linear case is that the 'rain gutter' need not be (and most probably is not) linear, but curved in this case. There may in fact be other valleys or rain gutters disconnected from the one containing $\boldsymbol{u}$. One can also see that if $\boldsymbol{\mathcal{A}}$ has full rank, all points in the immediate neighborhood of $\tilde{\boldsymbol{w}} = \boldsymbol{u}$ give a learning error larger than 0, *i.e.* there is a simple minimum at $\boldsymbol{u}$.

Assume that the learning finds one of these valleys. A small weight decay will pick out the point in the valley with the smallest norm among all the points in the valley. In general it can not be proven that picking that solution is the best strategy. But, at least from a philosophical point of view, it seems sensible, because it is (in a loose sense) the solution with the smallest complexity—the one that Ockham would probably have chosen.

The value of a weight decay is more evident if there are small errors in the targets. In that case one can go through exactly the same line of arguments as for the linear case to show that a weight decay can improve generalization, and even with the same optimal choice (17) of $\lambda$. This is strictly true only for small errors (where the linear approximation is valid).

## 6 NUMERICAL EXPERIMENTS

A weight decay has been tested on the NetTalk problem [11]. In the simulations back-propagation derived from the 'entropic error measure' [12] with a momentum term fixed at 0.8 was used. The network had $7 \times 26$ input units, 40 hidden units and 26 output units. In all about 8400 weights. It was trained on 400 to 5000 random words from the data base of around 20.000 words, and tested on a different set of 1000 random words. The training set and test set were independent from run to run.

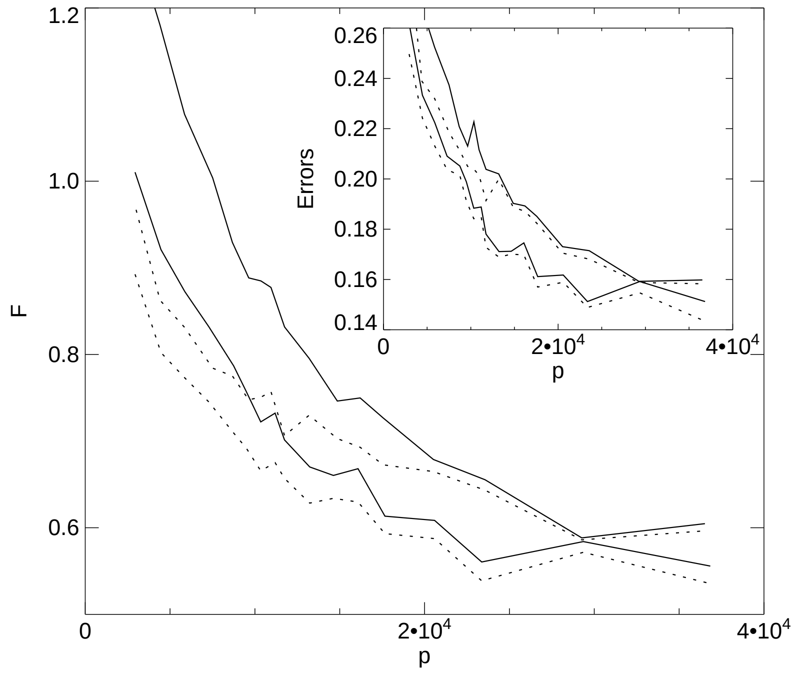



Figure 2: The top full line corresponds to the generalization error after 300 epochs (300 cycles through the training set) without a weight decay. The lower full line is with a weight decay. The top dotted line is the lowest error seen during learning without a weight decay, and the lower dotted with a weight decay. The size of the weight decay was $\lambda = 0.00008$.
Insert: Same figure except that the error rate is shown instead of the squared error. The error rate is the fraction of wrong phonemes when the phoneme vector with the smallest angle to the actual output is chosen, see [11].

Results are shown in fig. 2. There is a clear improvement in generalization error when weight decay is used. There is also an improvement in error rate (insert of fig. 2), but it is less pronounced in terms of relative improvement. Results shown here are for a weight decay of $\lambda = 0.00008$. The values 0.00005 and 0.0001 was also tried and gave basically the same curves.

## 7 CONCLUSION

It was shown how a weight decay can improve generalization in two ways: 1) It suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem. 2) If the size is chosen right, a weight decay can suppress some of the effect of static noise on the targets. Static noise on the targets can be viewed as a model of learning an unrealizable function. The analysis assumed that the network could be expanded around an optimal weight vector, and therefore it is strictly only valid in a little neighborhood around that vector.

The improvement from a weight decay was also tested by simulations. For the NetTalk data it was shown that a weight decay can decrease the generalization error (squared error) and also, although less significantly, the actual mistake rate of the network when the phoneme closest to the output is chosen.


### Acknowledgements

AK acknowledges support from the Danish Natural Science Council and the Danish Technical Research Council through the Computational Neural Network Center (CONNECT).


### References


[1] D.B. Schwartz, V.K. Samalam, S.A. Solla, and J.S. Denker. Exhaustive learning. *Neural Computation*, 2:371–382, 1990.

[2] N. Tishby, E. Levin, and S.A. Solla. Consistent inference of probabilities in layered networks: predictions and generalization. In *International Joint Conference on Neural Networks*, pages 403–410, (Washington 1989), IEEE, New York, 1989.

[3] E.B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[4] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 598–605, (Denver 1989), Morgan Kaufmann, San Mateo, 1990.

[5] H.H. Thodberg. Improving generalization of neural networks through pruning. *International Journal of Neural Systems*, 1:317–326, 1990.

[6] D.H. Weigend, D.E. Rumelhart, and B.A. Huberman. Generalization by weight-elimination with application to forecasting. In R.P. Lippmann et al, editors, *Advances in Neural Information Processing Systems*, page 875–882, (Denver 1989), Morgan Kaufmann, San Mateo, 1991.

[7] G.E. Hinton. Learning translation invariant recognition in a massively parallel network. In G. Goos and J. Hartmanis, editors, *PARLE: Parallel Architectures and Languages Europe. Lecture Notes in Computer Science*, pages 1–13, Springer-Verlag, Berlin, 1987.

[8] J .Moody. Generalization, weight decay, and architecture selection for nonlinear learning systems. These proceedings.

[9] D. MacKay. A practical bayesian framework for backprop networks. These proceedings.

[10] A. Krogh and J.A. Hertz. Generalization in a Linear Perceptron in the Presence of Noise. To appear in *Journal of Physics A* 1992.

[11] T.J. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

[12] J.A. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.