Behaviour in 0 of the neural networks training cost

CYRIL GOUTTE *
Department of Mathematical Modelling, Technical University of Denmark, building 321, 2800 Lyngby, Denmark
E-mail: cg@imm.dtu.dk

August 22, 1997

Abstract. We study the behaviour in zero of the derivatives of the cost function used when training non-linear neural networks. It is shown that a fair number of first, second and higher order derivatives vanish in zero, validating the belief that 0 is a peculiar and potentially harmful location. These calculations are related to practical and theoretical aspects of neural networks training.

Key words: training cost derivatives, neural networks training, numerical optimisation, regularisation

1. Introduction

Most algorithms used for training neural networks models minimise a cost function with multi-dimensional optimisation algorithms using derivatives information. Common examples are steepest descent, stochastic gradient descent [6, 4], conjugate gradient [5] or second order methods [1].

We will focus our attention on the behaviour of the cost function in 0. In that purpose, we derive in sections 3 to 5 the expressions of the derivatives of arbitrary order. These rather straightforward expressions show that a large number of the derivatives are zero when all parameters are zero, with the possible exception of the output bias.

We discuss the implications of these expressions, particularly with regards to initialisation, numerical optimisation, data normalisation and regularisation, in section 6.

2. Setting

We consider the case of a one-hidden-layer Multi-Layer Perceptron (MLP) model, i.e. a neural network containing one input layer, one layer of hidden "cells" implementing possibly non-linear transfer functions and one output layer with linear transfer function. Without loss of generality, we limit ourselves to only one output, one hidden cell and

* supported by a research grant from the Technical University of Denmark.

a multi-dimensional input. We will show in Section 3 that the use of one hidden cell is sufficient for our purpose.

We study the following model :

$$f_{\mathcal{W}}(\mathbf{x}) = W\, h(\mathbf{w}.\mathbf{x} + w_0) + W_0 \tag{1}$$

where $\mathbf{w}$ and $\mathbf{x}$ are P-dimensional vectors, $h(\cdot)$ is a (possibly non-linear) transfer function, and '.' indicates the scalar product between two vectors of the same size. $\mathcal{W}$ indicates the weight vector containing all the parameters of the model. The transfer function h is taken odd, i.e. symmetric around 0, and sufficiently differentiable so that all the derivatives used below can be calculated. A result of our assumptions is that $h(0) = h''(0) = 0$.

Training is performed by minimising the empirical risk [8]. For a quadratic risk, the cost function to optimise is:

$$C(\mathcal{W}) = \frac{1}{N} \sum_{k=1}^{N} \left(y^{(k)} - f_{\mathcal{W}}\left(\mathbf{x}^{(k)}\right) \right)^2 \tag{2}$$

where $\mathcal{D} = \left(\mathbf{x}^{(k)}, y^{(k)}\right)$ is an N element input-output sample. The choice of the quadratic cost is common for regression problems, and is consistent with our choice of a linear transfer function in the output.

By linearity of the differential operator, it is sufficient to analyse the derivatives of the local risk $R(\mathcal{W}) = (y - f_{\mathcal{W}}(\mathbf{x}))^2$. The expression of these derivatives will be derived in the next two sections. We will later discuss other risks (absolute value, cross-entropy) in section 5.

3. First and second derivatives

The first derivative of the cost w.r.t. a generic variable v is:

$$\frac{\partial R}{\partial v} = -2\,(y - f_{\mathcal{W}}(\mathbf{x}))\, \frac{\partial f_{\mathcal{W}}}{\partial v} \tag{3}$$

The first derivatives of the model w.r.t. the different parameters are:

$$\frac{\partial f_{\mathcal{W}}}{\partial W_0} = 1 \tag{4a}$$

$$\frac{\partial f_{\mathcal{W}}}{\partial W} = h(\mathbf{w}.\mathbf{x} + w_0) \tag{4b}$$

$$\frac{\partial f_{\mathcal{W}}}{\partial w_0} = W\, h'(\mathbf{w}.\mathbf{x} + w_0) \tag{4c}$$

$$\frac{\partial f_{\mathcal{W}}}{\partial \mathbf{w}} = W\, h'\left(\mathbf{w}.\mathbf{x} + w_0\right)\, \mathbf{x} \tag{4d}$$

Clearly, when all parameters are zero the derivatives of the model w.r.t. variables $\mathbf{w}$, w_0 or W vanish, making the derivatives of the model zero according to (3). On the other hand, in this situation (4a) leads to the following derivative of the cost function: $\frac{\partial C}{\partial W_0} = \overline{y}$. For centred data, this derivative is also zero. It is not when the data are not centred, but a single minimisation along W_0 sets the output bias to $W_0 = \overline{y}$, after which all derivatives are zero. W_0 is similar to a linear intercept.

The above equations also show why we do not lose any generality by using only one hidden unit. It is clear indeed that the derivatives w.r.t. a parameter involve only the parameters connected to the same hidden unit. If we did use several hidden cells, further derivatives w.r.t. parameters connected to other hidden units would be zero anyway.

The fact that all first derivatives are zero means that any training algorithm based on first order gradient information will stop. This does not mean however that a minimum has been reached, unless the Hessian is shown to be positive definite.

Second derivatives

The second derivative w.r.t. generic variables u and v is easily obtained:

$$\frac{\partial^2 R}{\partial v \partial u} = 2\left[\frac{\partial f_{\mathcal{W}}}{\partial u}\frac{\partial f_{\mathcal{W}}}{\partial v} - \left(y - f_{\mathcal{W}}\left(\mathbf{x}\right)\right)\frac{\partial^2 f_{\mathcal{W}}}{\partial v \partial u}\right] \tag{5}$$

According to the previous section, the first term in the square brackets is zero, except possibly when u and v are both W_0. The rest lies in the Hessian of the model. Deriving (4b) and (4a) we get:

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial W \partial W} = \frac{\partial^2 f_{\mathcal{W}}}{\partial W \partial W_0} = \frac{\partial^2 f_{\mathcal{W}}}{\partial W_0 \partial W_0} = \frac{\partial^2 f_{\mathcal{W}}}{\partial w_0 \partial W_0} = \frac{\partial^2 f_{\mathcal{W}}}{\partial \mathbf{w} \partial W_0} = 0 \tag{6}$$

The derivatives among the first layer parameters are:

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial {w_0}^2} = W\, h''\left(\mathbf{w}.\mathbf{x} + w_0\right) \tag{7a}$$

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial \mathbf{w} \partial w_0} = W\, h''\left(\mathbf{w}.\mathbf{x} + w_0\right)\, \mathbf{x} \tag{7b}$$

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial \mathbf{w}^2} = W\, h''\left(\mathbf{w}.\mathbf{x} + w_0\right)\, \mathbf{x} \otimes \mathbf{x} \tag{7c}$$

where $\otimes$ denotes the outer product. When $W = 0$, this portion of the Hessian is thus zero. The only terms left are the derivatives of (4c) and

(4d) w.r.t the output parameter W. These derivatives do not vanish in zero:

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial W \partial w_0} = h'(\mathbf{w}.\mathbf{x} + w_0) = h'(0) \tag{8a}$$

$$\frac{\partial^2 f_{\mathcal{W}}}{\partial W \partial \mathbf{w}} = h'(\mathbf{w}.\mathbf{x} + w_0)\ \mathbf{x} = h'(0)\ \mathbf{x} \tag{8b}$$

We know however that when all parameters (except possibly the output bias) are zero, $f_w(x) = W_0 = \overline{y}$ after at most one line minimisation. Accordingly, $\frac{\partial^2 C}{\partial W \partial w_0} = (\overline{y} - W_0)\ h'(0) = 0$. On the other hand, no such simplification is possible for (8b). The expression of the second derivative turns out as:

$$\frac{\partial^2 C}{\partial W \partial \mathbf{w}} = -2h'(0)\,\sigma_{\mathbf{x}y} \tag{9}$$

where $\sigma_{\mathbf{x}y}$ is the vector of input-output covariances. In the general case, covariances can be positive as well as negative. An interesting situation arises in the case where $\sigma_{\mathbf{x}y} = 0$. This can occur when we model only the strictly non-linear part of the input-output mapping. This is the case when we subtract the linear estimation from the measured output, or alternatively using additional direct connections from the input to the output. In that case, the Hessian of the cost function is uniformly zero when all parameters are zero.

These derivations show that zero is indeed not a local minimum of the cost function. Only a small fraction of the Hessian is possibly different from zero, and the Hessian is not positive definite, suggesting that zero is a saddle point of the cost surface.

4. Higher order derivatives

We will now give general expressions of higher order derivatives, using the following notations. $\mathcal{V} = (v_i)$ is a collection of P variables taken among the parameters of our model. The derivative of function f w.r.t. these variables will be written:

$$\frac{\partial^P f}{\partial \mathcal{V}} = \frac{\partial^P f}{\prod_{i=1}^{P} \partial v_i} \tag{10}$$

Furthermore, $\mathcal{P}(\mathcal{V})$ is the set of all possible partitions of $\mathcal{V}$, i.e. couples (s_1, s_2) such that $s_1 \cup s_2 = \mathcal{V}$. $\mathcal{P}(\mathcal{V})^*$ is the set of partitions where neither s_1 nor s_2 is empty. Notice that strictly speaking we are not considering sets as replications of variables are possible. In these conditions, we introduce the following lemma (proven in appendix A):

LEMMA 1 (Cost derivatives). *The P-th derivative of the quadratic cost function with respect to a collection $\mathcal{V}$ of P parameters is:*

$$\frac{\partial^P R}{\partial \mathcal{V}} = 2 \left[\sum_{(s_1,s_2)\in\mathcal{P}(\mathcal{V})^*} \frac{\partial f_{\mathcal{W}}}{\partial s_1} \frac{\partial f_{\mathcal{W}}}{\partial s_2} - (y - f_{\mathcal{W}}(\mathbf{x})) \frac{\partial^P f_{\mathcal{W}}}{\partial \mathcal{V}} \right] \tag{11}$$

According to this lemma, cost derivatives are composed of only two terms involving derivatives of the model. Such derivatives are given by the second lemma:

LEMMA 2 (Model derivatives). *The P-th derivative of the model mapping $f_{\mathcal{W}}(x)$ are the following:*

(a) For a collection $\mathcal{V}$ of $P > 1$ first layer parameters (i.e. w_0 or $w_j \in w$), there exist α independent of the parameters such that:

$$\frac{\partial^P f_{\mathcal{W}}}{\partial \mathcal{V}} = \alpha\, W\, h^{(P)}(\mathbf{w}.\mathbf{x} + w_0) \tag{12}$$

(b) With similar notation, for $P-1$ first layer parameters:

$$\frac{\partial^P f_{\mathcal{W}}}{\partial W \partial \mathcal{V}} = \alpha\, h^{(P-1)}(\mathbf{w}.\mathbf{x} + w_0) \tag{13}$$

(c) all other P-th derivatives are zero.

This leads to the following proposition:

PROPOSITION 1. *For an odd, infinitely derivable transfer function h, all odd-order derivatives of the quadratic cost function are zero when the output bias is $\overline{y}$ and the other parameters are all zero.*

Proof. Shown in section 3 for $P = 1$. Trivial from the previous lemmas for $P > 1$, as all terms of the derivative (11) include an odd-order derivative of the model, which by lemma 2 is zero in these parameter conditions $\square$

For even-order derivatives, the first term in the right-hand side of 11 contains even derivatives, in which case the result is potentially nonzero, as exemplified in section 3 for P=2.

5. Other risk functions

The previous section is relevant to the use of a quadratic risk function. Let us analyse the results obtained with other risks. Let us first notice that lemma 2 is still valid, so that a number of derivatives of the *model* are zero, notably odd order derivatives.

Absolute value error

A common choice in regression is the *absolute value error* $R(\mathcal{W}) = |y - f_{\mathcal{W}}(\mathbf{x})|$, which enjoys some robustness properties. In our framework, the main problem with this risk is that strictly speaking it has no derivative in 0. It can however be approximated by the sign function as long as $y \neq f_{\mathcal{W}}(\mathbf{x})$, giving the first derivative w.r.t. a generic parameter v:

$$\frac{\partial R}{\partial v} = -\mathrm{sign}\left(y - f_{\mathcal{W}}(\mathbf{x})\right) \frac{\partial f_{\mathcal{W}}}{\partial v} \tag{14}$$

According to the model derivatives calculated in section 3, this is zero when all parameters are zero, with the possible exception of the output bias. This bias takes the value $W_0 = \mathrm{median}(y)$ in at most one iteration.

Equation 14 suggests a modified version of lemma 1. As the sign function is piecewise constant, the expression of higher order derivatives of the cost function becomes:

$$\frac{\partial^P R}{\partial \mathcal{V}} = -\mathrm{sign}\left(y - f_{\mathcal{W}}(\mathbf{x})\right) \frac{\partial^P f_{\mathcal{W}}}{\partial \mathcal{V}} \tag{15}$$

provided that $y \neq f_{\mathcal{W}}(\mathbf{x})$. Conclusions from the previous section are thus unchanged: all odd-order derivatives will vanish when all parameters are zero, and the output bias is the median of the data outputs.

Cross-entropy

A popular choice of error function for classification problems is the cross-entropy $R(\mathcal{W}) = (1+y)\ln(1+f_{\mathcal{W}}(\mathbf{x})) + (1-y)\ln(1-f_{\mathcal{W}}(\mathbf{x}))$ for -1/+1 classification. This function is infinitely derivable in the domain where $f_{\mathcal{W}}(\mathbf{x}) \neq \pm 1$. The first derivative of the cost function w.r.t. a generic parameter v is given by:

$$\frac{\partial R}{\partial v} = \frac{2\left(y - f_{\mathcal{W}}(\mathbf{x})\right)}{1 - f_{\mathcal{W}}(\mathbf{x})^2} \frac{\partial f_{\mathcal{W}}}{\partial v} \tag{16}$$

According to the model derivatives calculated in section 3, this is zero when all parameters are zero, except for the output bias which will again take the mean value of the output.

Contrary to the case of quadratic risk, the derivatives of the cross-entropy w.r.t. the estimation $f_{\mathcal{W}}(\mathbf{x})$ do not vanish after a certain order.

This means that the convenient (11) from lemma 1 does not hold. Instead the expression of higher order derivatives of the cost function include higher derivatives of the risk function as well as combinations of derivatives of the model.

A general expression equivalent to (11) can be derived but the resulting formula is tedious and will not be presented here. The general conclusion, however, is unchanged. The odd-order derivatives of the cross-entropy risk vanish when all parameters are zero, and the output bias is the average of the data outputs.

6. Discussion

We will now evoke the relevance of these results to a number of topics associated with the theoretical or numerical aspects of training.

First these results confirm that parameters should not be initialised to zero prior to training of the model by numerical optimisation. It has been shown however that the Hessian is not positive, so that 0 is not an attractor. Even for small deviation from zero, the parameters will tend to diverge from zero and eventually reach a local minimum.

This is especially relevant when one considers the effect of linear transformations of the data such as normalisation. Let us consider the raw data $(\mathbf{x}, y)$ and the normalised data $(\widetilde{\mathbf{x}}, \widetilde{y})$, which are assumed to be a factor a smaller, i.e. $\mathbf{x} = a\,\widetilde{\mathbf{x}}$ and $y = a\,\widetilde{y}$. It is usually assumed that training on large, raw data requires to initialise the input parameters to small values. This avoids possible saturation of the transfer function, a situation that can hinder or even stop the optimisation. The best parameters for one type of data is easily obtained from the best parameters for the other by a linear transformation: $W = a\,\widetilde{W}$ and $w = \widetilde{w}/a$ for non-bias parameters. This suggest that the range of the initialisation will be proportional (resp. inversely proportional) to a for output (resp. input) parameters. It could be feared from the derivations of the previous chapter that large data would thus lead to initial conditions dangerously close to zero, where training is inefficient. However, as mentioned in the previous paragraph, the parameters are guaranteed to diverge from this region. Furthermore, the derivatives of the cost function scale with a. This indicates that for parameters initialised close to 0 (i.e. large a), the gradient will be large so that the optimisation will easily extract the parameters from a potentially difficult zone.

The above results also bear some significance to the numerical optimisation that is used for training. It is quite clear indeed that zero is potentially harmful to methods using first-order information to minimise the cost function. This is the case for the bulk of the methods used for training neural networks, batch or stochastic gradient descent, conjugate gradient or quasi-Newton methods, as well as Levenberg-Marquart. Training using any of these methods will stop when parameters happen to take zero values. However, provided initialisation has been performed properly, this is a very unlikely situation. Passing in the neighbourhood of 0 during optimisation will have the same effect as the initialisation to small values mentioned above.

Regularisation

Sensible training of neural networks models will usually not minimise the empirical cost, as this leads to overly optimistic models, a phenomenon known as over-fitting. A popular and widespread remedy is the use of regularisation [3]. A number of regularisation techniques have been proposed [2], which involve optimising a regularised cost consisting of the empirical risk and an additional, positive regularisation term.

Regularisation terms using the parameters directly rely on the assumption that small weights correspond to smooth mapping, and are thus minimal when parameters are zero. This means that the first derivative of the regulariser w.r.t. parameters is zero in zero. As regularised cost is the sum of the empirical risk studied in sections 3 and 4 and this regulariser, its first order derivatives will be zero two. Higher-order derivatives will also be zero for the most common regularisers. E.g. the weight-decay term is quadratic in the weights so that derivatives of orders greater than 2 disappear. The regularisation term will thus have no influence on higher-order derivatives of the regularised cost function.

The well-known Tikhonov regulariser [7] involve squared derivatives of the model, imposing a direct constraint on the smoothness of the model. It should be clear from previous sections that the squared derivatives of the model when parameters are zero is zero, and thus a minimum of the regulariser. The same argument as above applies to show that the first derivatives of the regularised cost function are also zero in those conditions. For general non-linear models however, higher order derivatives of the Tikhonov regulariser are potentially non-zero as they involve higher-order, even derivatives of the model that are not zero as shown earlier.

7. Conclusion

Neural networks models are usually trained with non-linear optimisation methods using first order derivatives. It is therefore interesting to study the behaviour of the derivatives of the cost, including higher-order derivatives. We have shown here that in the situation where all parameters are zero, except possibly the output bias, the cost function used for training the model has a number of vanishing derivatives. This characterises the behaviour of the cost in the close neighbourhood of zero and confirms that it is a peculiar and potentially harmful location for the weight vector. This suggests a number of remarks with regards to numerical aspects of neural networks training, as well as the use of a number of common regulariser.

Acknowledgements

The author wishes to acknowledge discussions with Lars Kai Hansen and Jan Larsen.

References

1. R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method", Neural Computation, vol. 4, no. 2, pp. 141–166, March 1992.
2. C. Goutte, Statistical learning and regularisation in regression, Ph.D. thesis, Université Paris 6: Paris, 1997.
3. A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization", in J. E. Moody, S. J. Hanson and R. P. Lippman (eds.), Advances in Neural Information Processing Systems, vol. 4, 1992.
4. Y. LeCun, Modèles connexionnistes de l'apprentissage, Ph.D. thesis, Université Paris 6: Paris, 1987.
5. M. Møller, "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks, vol. 6, no. 4, pp. 525–533, 1993.
6. D. Rumelhart, G. Hinton and R. Williams, "Learning internal representation by error propagation", in D. Rumelhart and J. McClellan (eds.), Parallel Distributed Processing : exploring the microstructure of cognition, vol. 1, pp. 318–362, MIT Press, 1986.
7. A. N. Tikhonov and V. Y. Arsenin, Solution of Ill-Posed Problems, Winston: Washington, D.C., 1977.
8. V. N. Vapnik, The Nature of Statistical Learning Theory, Springer: New York, 1995.

Appendix

A. Proof of Lemma 1

Proof. By induction on the partitions of the parameters. The first two steps are proven in section 3. Let us suppose that (11) holds for a collection $\mathcal{V}$ of P parameters, and introduce an additional variable t. Differentiating (11) w.r.t. this new variable:

$$\frac{\partial^{P+1}R}{\partial t \partial \mathcal{V}} = 2\left[\sum_{(s_1,s_2)\in\mathcal{P}(\mathcal{V})^*}\left(\frac{\partial f_{\mathcal{W}}}{\partial t \partial s_1}\frac{\partial f_{\mathcal{W}}}{\partial s_2} + \frac{\partial f_{\mathcal{W}}}{\partial s_1}\frac{\partial f_{\mathcal{W}}}{\partial t \partial s_2}\right) + \frac{\partial f_{\mathcal{W}}}{\partial t}\frac{\partial f_{\mathcal{W}}}{\partial \mathcal{V}} - \left(y - f_{\mathcal{W}}(\mathbf{x})\right)\frac{\partial^{P+1} f_{\mathcal{W}}}{\partial t \partial \mathcal{V}}\right] \quad (17)$$

Now define $\mathcal{U} = \mathcal{V} \cup \{t\}$, and show that the sum of all the terms in the first line of (17) is generated by the sum over all elements of $\mathcal{P}(\mathcal{U})^*$.

Clearly, $(\{t\} \cup s_1, s_2)$, $(s_1, \{t\} \cup s_2)$ and $(\{t\}, \mathcal{V})$ are all members of $\mathcal{P}(\mathcal{U})^*$. Reciprocally, consider an element (u_1, u_2) of $\mathcal{P}(\mathcal{U})^*$. With no loss of generality, we will suppose that t is in u_1. Define $s_2 = u_2$, a non-empty partition of $\mathcal{V}$, and $s_1 = u_1 \backslash t$. If s_1 is non-empty then (s_1, s_2) is a member of $\mathcal{P}(\mathcal{V})^*$, so that the corresponding cross-derivative $\frac{\partial f_{\mathcal{W}}}{\partial u_1}\frac{\partial f_{\mathcal{W}}}{\partial u_2}$ is under the sum on the top line of (17). Otherwise, $u_1 = \{t\}$ and $u_2 = \mathcal{V}$ so that the cross-derivative is the leftmost term on the same line. Equation 17 can thus be re-written as:

$$\frac{\partial^{P+1}R}{\partial \mathcal{U}} = 2\left[\sum_{(u_1,u_2)\in\mathcal{P}(\mathcal{U})^*}\frac{\partial f_{\mathcal{W}}}{\partial u_1}\frac{\partial f_{\mathcal{W}}}{\partial u_2} - \left(y - f_{\mathcal{W}}(\mathbf{x})\right)\frac{\partial^{P+1} f_{\mathcal{W}}}{\partial \mathcal{U}}\right] \quad (18)$$

which completes the proof □

B. Proof of Lemma 2

Proof. By induction on the order of the derivatives. The first step ($P = 2$) is proven in section 3. Let us suppose that lemma 1 holds for $P \geq 1$. Each P-th order derivative can be differentiated by one of W_0, W, or v, where v is any first layer parameter (either w_0 or one of the $w_j \in w$). The following table shows how each of the three cases of lemma 1 for $P+1$ are obtained by differentiating the cases of lemma 1 for P. E.g. a derivative given by case (b) of lemma 1 for P differentiated w.r.t. W leads to a derivative given by case (c) of lemma 1 for $P+1$.

	$/\partial v$	$/\partial W$	$/\partial W_0$
∂ (a)	(a)	(b)	(c)
∂ (b)	(b)	(c)	(c)
∂ (c)	(c)	(c)	(c)

The last row and last column are obvious as neither (12) nor (13) depend on W_0. Similarly, differentiating (13) w.r.t. W will give zero. On the other hand, differentiating (13) w.r.t. a first layer parameter gives:

$$\frac{\partial}{\partial w_0}\left(\alpha\, h^{(P-1)}\left(\mathbf{w}.\mathbf{x}+w_0\right)\right)=\alpha\, h^{(P)}\left(\mathbf{w}.\mathbf{x}+w_0\right) \tag{19a}$$

$$\frac{\partial}{\partial w_j}\left(\alpha\, h^{(P-1)}\left(\mathbf{w}.\mathbf{x}+w_0\right)\right)=\alpha\, x_j\, h^{(P)}\left(\mathbf{w}.\mathbf{x}+w_0\right) \tag{19b}$$

Both expressions are of the same form as (13), with different values of α, which completes the proof for the second row of the table. Let us now consider (12). Clearly, differentiating it w.r.t. W will give an expression of the same form as (13), where α is unchanged.

The only remaining case is that of a derivative involving only first layer parameters. As W is independent of such parameters, $\alpha.W$ now plays the same role as α in (19a-19b), leading to two expressions of the same form as (12). This achieves the proof of the inductive step, and thus that of lemma 1 □