Instantaneous learning algorithms

- Many ANNs can aproximate arbitrarily well any continuous linear function, with network's architecture and learning algorithm determining how easily learning proceeds;
- For on-line modelling and control, it is unrealistic to assume that the input signal will excite the whole of the state-space, and so the effects of a reduced input signal on the overall functional approximation must be investigated;
- Hence, learning must be local, i.e., the parameters adapted should only affect the output of the network locally;
- Instantaneous learning rules are formulated by minimising instantaneous estimates of a performance function, which is generally the MSE output error and the parameters are updated using gradient descent rules;

- The AMNs depend linearly on a set of weights and therefore instantaneous learning laws are appropriate for estimating these weights. Learning is local due to the characteristics of the basis functions.
- An important issue is the relationship between instantaneous and true gradient descent. The instantaneous estimates made of the gradient introduces noise into the updating algorithm, causing erratic weight updates, and when modelling error exists, the parameters converge to a minimal capture zone.
- The size of the gradient noise is related to the condition of the autocorrelation matrix, providing a direct link with the rate of convergence of the true gradient descent.

Instantaneous Gradient Descent

The MSE of the network is given by:

$$J = E(\epsilon_y^2(t)), \tag{1}$$

where $\epsilon_y(t)$ is the a priori output error $(\hat{y}(t) - y(t))$, and $E(.)$ is the expectation operator.

The instantaneous estimate of the MSE at time t is:

$$J_i(t) = \epsilon_y^2(t), \tag{2}$$

and the instantaneous estimate of the gradient of the performance function at time t is given by:

$$\frac{\partial J_i(t)}{\partial \boldsymbol{w}} = -2\epsilon_y(t)\boldsymbol{a}(t) \tag{3}$$

Instantaneous gradient descent training rules update the weight vector in proportion to the negative instantaneous gradient, producing a learning rule of the form:

$$\Delta \boldsymbol{w}(t-1) = \delta \epsilon_y(t)\boldsymbol{a}(t), \tag{4}$$

where $\Delta \boldsymbol{w} = \boldsymbol{w}(t) - \boldsymbol{w}(t-1)$.

- The search path is parallel to the transformed input vector and the step size is equal to the learning rate multiplied by the instantaneous output error.
- This is the well known LMS instantaneous gradient descent rule, which has the basic form:

$$\boxed{\text{weight update} = \text{scalar} * \text{transformed input vector}}$$

- If any of the weights in a AMN have not been initialised then it is possible to use:

$$w_i(t) = \begin{cases} \hat{y}(t) & \text{if the } i\text{-th weight was not initialised and, } a_i(t) > 0 \\ w_i(t-1) + \delta \epsilon_y(t) a_i(t) & \text{otherwise} \end{cases} \quad (5)$$

Performance Surface

An instantaneous peformance surface generated by a single training is shown in the Figure below.

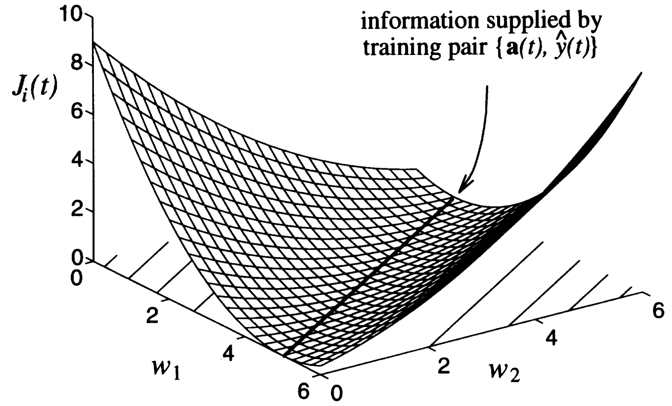

Figure 5.1 An instantaneous performance surface in two-dimensional weight space. At time t, the information supplied to the network is contained in the data pair: $\mathbf{a}(t) = (0.6, 0.4)$, $\hat{y}(t) = 3$.

Each training pair presented to the network produces one equation which the p weights must satisfy:

$$\boldsymbol{a}^T(t)\boldsymbol{w} = \hat{y}(t), \tag{6}$$

in order to store the data exactly.

Hence the performance surface is singular, as many weight vectors satisfy this equation and the instantaneos optimisation problem is underconstrained.

When the training information is rich enough, or the input is suficiently exciting, the global minima of sucessive performance surfaces intersect at a unique point which is the gloabl minimum ot the true MSE performance function as ilustrated below.

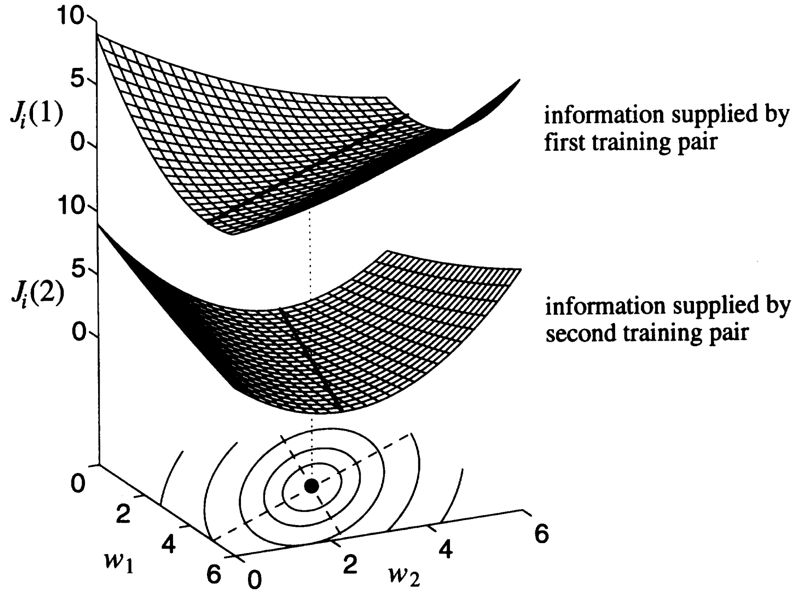

Figure 5.2 The performance surfaces generated by the two training pairs: $\mathbf{a}(1) = (0.6, 0.4)$, $\widehat{y}(1) = 3$ and $\mathbf{a}(2) = (0.2, 0.8)$, $\widehat{y}(2) = 3$. The global minima of each of these performance surfaces (bold lines) are projected onto the base of the figure (dashed lines) where they intersect at a unique point. The contour plot of the average or expected MSE is also shown.

The instantaneous autocorrelation matrix $\boldsymbol{R}(t)$ associated with the $J_i(t)$ is of the form:

$$\boldsymbol{R}(t) = \boldsymbol{a}(t)\boldsymbol{a}^T(t) \tag{7}$$

This a $p \times p$ matrix which has only one non-zero eigenvalue ($\| \boldsymbol{a}(t) \|_2^2$) and the corresponding eigenvector is $\boldsymbol{a}(t)$.

Therefore, the curvature of the instantaneous performance surface along the major axis, $\boldsymbol{a}(t)$, is $2 \| \boldsymbol{a}(t) \|_2^2$ and is flat along all the remaining axes.

Learning stability is assured if:

$$0 < \delta < \frac{2}{\| \boldsymbol{a}(t) \|_2^2} \tag{8}$$

Output Error Reduction

$\lvert\epsilon_{\underline{y}}(t)\rvert > \lvert\epsilon_y(t)\rvert$	if $\delta \notin [0, 2/ \parallel \boldsymbol{a}(t) \parallel_2^2]$
$\lvert\epsilon_{\underline{y}}(t)\rvert = \lvert\epsilon_y(t)\rvert$	if $\delta = 0$ or $\delta = 2/ \parallel \boldsymbol{a}(t) \parallel_2^2$
$\lvert\epsilon_{\underline{y}}(t)\rvert < \lvert\epsilon_y(t)\rvert$	if $\delta \in (0, 2/ \parallel \boldsymbol{a}(t) \parallel_2^2)$
$\lvert\epsilon_{\underline{y}}(t)\rvert = 0$	if $\delta = 1/ \parallel \boldsymbol{a}(t) \parallel_2^2$

Normalised Least Mean Squares

Based on the fact that:

$$|\epsilon_{\underline{y}}(t)| = 0$$

when:

$$\delta = 1/ \parallel \boldsymbol{a}(t) \parallel_2^2$$

It is possible to derive a Normalised LMS which is of the following form:

$$w_i(t) = \begin{cases} \hat{y}(t) & \text{if the } i\text{-th weight was not initialised and, } a_i(t) > 0 \\ w_i(t-1) + \frac{\delta \epsilon_y(t)}{\|\boldsymbol{a}(t)\|_2^2} a_i(t) & \text{otherwise} \end{cases} \tag{9}$$

Normalised MSE performance function

- The output error reduction does not depend on the magnitude of the transformed input vector when the weight vector is trained using the NLMS learning rule;
- However, the dependence of the learning rate on the size of the transformed input vector means that the NLMS rule no longer minimizes the MSE but a normalised MSE performance function:

$$J = E\left(\frac{\epsilon_y^2(t)}{\| \boldsymbol{a}(t) \|_2^2}\right). \tag{10}$$

- If there exists a unique weight vector for which $\epsilon_y(t) = 0 \ \forall t$ or if $\| \boldsymbol{a}(t) \|_2 = cte, \forall t$, the optimal weight vectors are equivalent.

Geometric Interpretation of the LMS rules

- the geometric interpretation was first proposed by Kaczmarz (1937), when he proposed the NLMS rule for solving a consistent set of simultaneous linear equations;
- The LMS and the NMLS rule have the same basic form:

$$\Delta \boldsymbol{w}(t-1) = c(t)\boldsymbol{a}(t). \tag{11}$$

- The desired output of the network at time t is $\hat{y}(t)$ and thus the aim is to update the weight vector $\boldsymbol{w}(t-1)$ so that:

$$\hat{y}(t) = \boldsymbol{a}^T(t)\boldsymbol{w}(t); \tag{12}$$

- the solution to this equation lies on the $(p-1)$-dimensional solution hyperplane $\boldsymbol{h}(t)$ which is composed of the set of solutions to the following equation:

$$0 = \boldsymbol{a}^T(t)\boldsymbol{w}(t) - \hat{y}(t); \tag{13}$$

Geometric interpretation of the LMS rules

The search direction for the LMS rule is parallel to the transformed input vector and the transformed input vector is perpendicular to the solution hyperplane, which means that the weight change is also perpendicular to the solution hyperplane as shown in the Figure below:

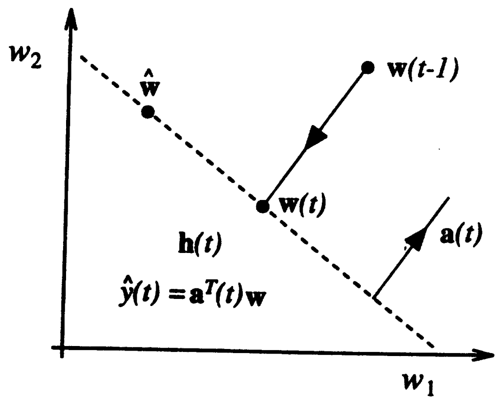

Figure 5.3 A solution hyperplane (dashed line) and perpendicular learning in two-dimensional weight space.

The effect of different values of δ used in NLMS

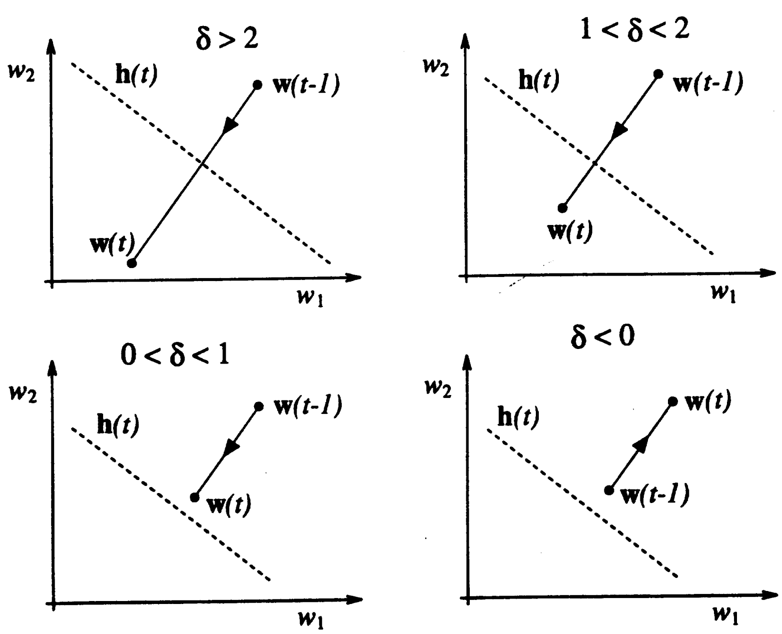

Figure 5.4 The effect on the weight updates for different values of δ in the NLMS adaptation rule.

Principle of minimal disturbance

- The LMS family of learning rules embody the principle of minimal disturbance, that is to say, the change made to the weight vector is the smallest which causes the new desired output to be stored;

- Since the weight change is the smallest of all the possible changes which the new weight vector to store the new desired output, it interferes minimally with the learnt information.

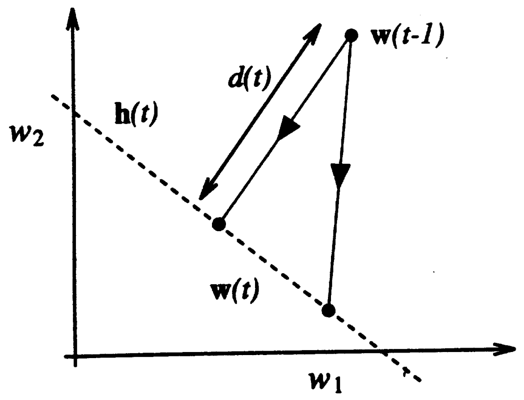

Figure 5.5 The principle of minimal disturbance for LMS learning rules.

Weight update using NLMS

- the function to be approximated:

$$\hat{y}(t) = -0.5 + x(t).$$

- Model 1:
  - $y(t) = a_1(x(t))w_1 + a_2(x(t))w_2$
    $a_1 = 1.0 \qquad a_2 = x(t)$
  - $\delta = 1.0$
  - Normalised condition number: $C(\boldsymbol{R}) = 18.6$
  - Initial value: $\boldsymbol{w}(0) = (-1.0, 0.0)^T$
  - Converges ($\| \epsilon_w(t) \|_\infty < 10^{-6}$) after 270 iterations.
- Model 2:
  - $y(t) = a_1(x(t))w_1 + a_2(x(t))w_2$
    $a_1 = 1.0 - x(t) \qquad a_2 = x(t)$
  - $\delta = 1.0$
  - Normalised condition number: $C(\boldsymbol{R}) = 3.66$
  - Initial value: $\boldsymbol{w}(0) = (0.0, -0.5)^T$
  - Converges ($\| \epsilon_w(t) \|_\infty < 10^{-6}$) after 50 iterations.

Example: weight update using NLMS

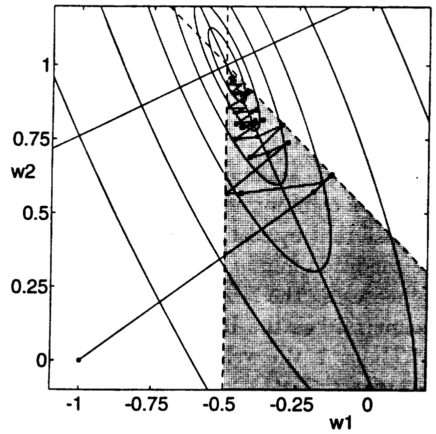

Figure 5.6 Weight convergence for a linear model trained using the NLMS rule. The contours are given by $\xi = 0.001, 0.003, 0.01, 0.03, 0.1, 0.3$. The zone of convergence (spanned by the solution hyperplanes) is the shaded area.

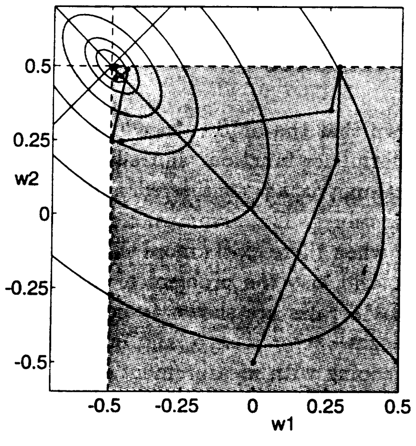

Figure 5.7 Weight convergence for a B-spline model trained using the NLMS rule. The contours are given by $\xi = 0.001, 0.003, 0.01, 0.03, 0.1$, and the shaded area shows the zone of convergence.

Parameter convergence

Suppose that there is a set of training examples $\{\boldsymbol{x}(t), \hat{y}\}_{t=1}^{L}$, which is cyclically presented to an AMN trained using the NLMS rule.

Assuming that there exists a unique weight vector $\hat{w}$ such that:

$$\hat{y} = \boldsymbol{a}^T \hat{\boldsymbol{w}} \quad \forall t \tag{14}$$

Although this assumption is not reasonable it is possible to prove that under this conditions and if the training data is sufficiently rich, output convergence implies also implies parameter convergence

This is illustrated in the next Figure where the weight vector is projected orthogonally onto each solution hyperplane and because the solution hyperplanes intersect at a unique point (consistent data), convergence of the learning algorithm is obvious.

Parameter convergence

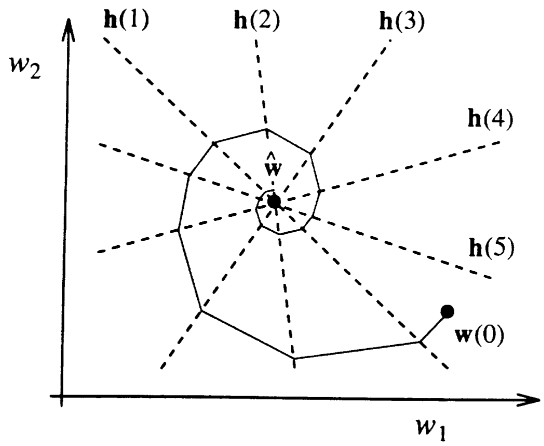

Figure 5.8 Parameter convergence for a consistent set of training data. Each training pair generates a solution hyperplane (dotted line) and because the set is consistent, they intersect at a unique point. The weight vector is projected orthogonally onto each solution hyperplane at each training iteration.

Weight and output convergence

- The fact that the size of the weight vector error and the instantaneous output error decrease for each update do not imply that the overall MSE decreases for each update;
- The MSE might increase due to the effect of the update on the information already stored in the weight vector;
- Storing one training example, causes a significant reduction in the information retained about neighbouring data pairs, and this effect is known as learning interference;
- The instantaneous estimates which are being formed of the gradient introduce noise into weight updates, and this can cause the overall MSE to increase.

Model mismatch and output dead-zones

- When there is modelling error or measurement noise output dead-zones can be used to counteract this behaviour partially;
- The technique does not update the weight vector when the output error is less than some predefined value ζ, and if it is greater, the difference is used in place of the output error;
- This can be written as:

$$\epsilon_y^d(t) = \begin{cases} 0 & \text{if } |\epsilon_y(t)| \le \zeta \\ \epsilon_y(t) + \zeta & \text{if } \epsilon_y(t) < -\zeta \\ \epsilon_y(t) - \zeta & \text{if } \epsilon_y(t) > \zeta \end{cases} \tag{15}$$

where ϵ_y^d is the modified output error used in place of the normal output error in the LMS learning rules.

Stochastic approximation

- As an alternative to counteracting model mismatch using output dead-zones, it is possible to filter out the gradient noise using a stochastic approximation LMS learning rule;
- The stochastic approximation LMS learning rule is the following:
  1. assign an individual learning rate to each basis function; and
  2. reduce δ_i, through time as the confidence in a particular weight increases.
- These modifications retain the fast initial convergence rate while in the long term filtering out measurement and modelling noise.

Rate of convergence

- The rate of convergence of the NLMS learning algorithm depends directly on the orthogonality of the training pattern;
- The rate of the parameter convergence depends on the relative orientations of the sucessive weight vector errors and the transformed input vectors;
- Using NLMS with $\delta = 1$ the learning rate is fast and there is no interference when the transformed input vectors are orthogonal.
- In the Figure below it is shown the convergence, after two iterations, a system with two weights and with the transformed inputs $\boldsymbol{a}(1)$ and $\boldsymbol{a}(2)$ after two iterations.

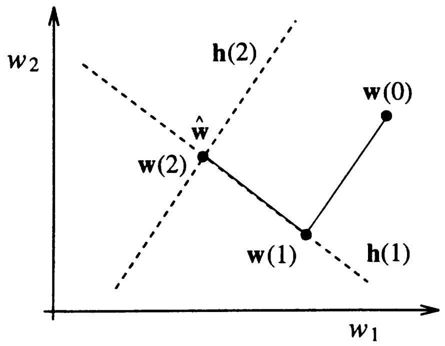

Figure 5.10 Orthogonal solution hyperplanes in two-dimensional weight space.

Slow convergence with correlated data

In contrast, if the transformed vectors are nearly parallel (i.e. when the set of the transformed input vector are ill-condiitoned) the convergence can be very slow.

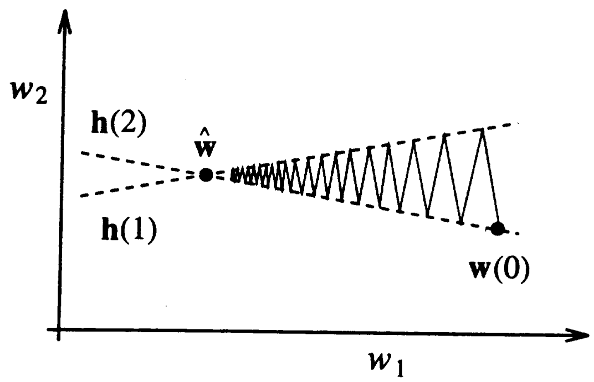

Figure 5.11 Ill-conditioned solution hyperplanes in a two-dimensional weight space.

Noisy instantaneous gradient estimates

- the minimal capture zones are simply due to the noisy estimates of the true gradient when instantaneous learning rules are used;
- The noise in the gradient estimate makes it impossible to determine when the current set of weights are optimal when modelling errors exists between the desired function and the network, and so adaptation does not stop, even if the current weight vector is optimal;
- The instantaneous covariance of the gradient noise is given by:

$$\boldsymbol{N} = \text{cov}(\epsilon_{\triangledown}) = E(\epsilon_{\triangledown}\epsilon_{\triangledown}^T).$$

- The covariance of the weight change can be written as:

$$\text{cov}(\Delta \boldsymbol{w}) = \delta^2 \boldsymbol{N}$$

Minimal capture zone

- when no optimal weight vector exists which can reproduce the training data exactly, the weights converge to a domain rather than to a unique value;
- The size and the shape of these minimal capture zones depends on the training samples, their order of presentation, the internal representation of the network, the adaptation rule and the learning rate δ;

A two-dimensional capture zone is shown in the Figure below:

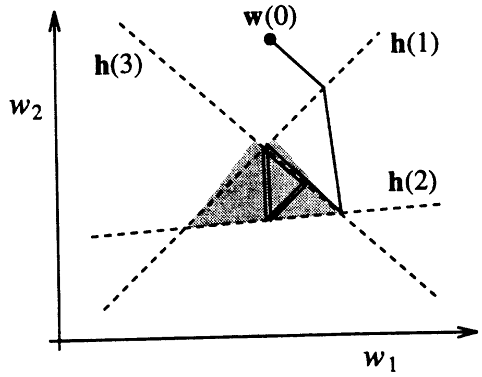

Figure 5.16 A two-dimensional minimal capture zone (shaded area) generated by three inconsistent solution hyperplanes. The weights no longer converge to an optimal value.