The Least-Mean-Square (LMS) algorithm and its geophysical applications

Brian Russell[1]

[1] Hampson-Russell, A CGG GeoSoftware Company, Calgary, Alberta, brian.russell@cgg.com

ABSTRACT

In geophysical analysis, we are familiar with the least-squares or Wiener-Levinson algorithm since is the basis for solving many of our most common processing problems. The least-squares method is applied to problems in which a set of unknown weights are extracted which transform a measured signal into a desired signal. To apply the least-squares algorithm the signal must first be fully acquired, which is standard practice in seismic acquisition. The least-mean-square (LMS) algorithm, on the other hand, is an adaptive filter that does not require knowledge of the full signal, but determines the weights adaptively on a sample by sample basis. The LMS algorithm was developed in the electrical engineering community and has found great success in such applications as the adaptive equalization of telephone channels and blood pressure regulation. It also lead to the development of the first feedforward neural networks, which were only able to solve linearly separable problems. The LMS algorithm was then was extended to feedforward networks that could solve nonlinearly separable problems. Thus, an understanding of the LMS algorithm is the first step in understanding neural networks and machine learning. In this study I will not specifically look at neural network algorithms but rather will focus the least-squares technique and gradient search methods such as steepest descent and conjugate gradient, and then see how LMS compares to these methods. To this this, I will use two straightforward examples, one from electrical engineering and one from geophysics.

The geophysical example will be the wavelet deconvolution problem. We will first solve wavelet deconvolution using the least-squares, steepest descent and conjugate gradient algorithms. These methods all converge to the correct result. We will then look at how the LMS algorithm works on the deconvolution problem. Although LMS does not converge completely in our example because of the lack of samples, it works much better on longer datasets and also potentially solves the problem of time-varying wavelets.

INTRODUCTION

The least-mean-square (LMS) algorithm is an adaptive filter that was first developed by Widrow and Hoff (1960). Along with the perceptron learning rule (Rosenblatt, 1962) this lead to the development of the first feedforward neural networks, which were able to solve linearly separable problems. The extension to feedforward networks that could solve nonlinearly separable problems by Rumelhart et al. (1986) also relied on LMS. Thus, an understanding of the LMS algorithm is the first step in understanding neural networks and machine learning. In this talk, I will first use an electrical engineering example from Widrow and Stearns (1985) to explain the LMS algorithm, and also discuss the full least-squares, gradient descent and conjugate gradient methods.

My second example comes from geophysics, in which I perform wavelet deconvolution (Claerbout, 1976). In this example, we will first solve the deconvolution problem using the least-squares, steepest descent and conjugate gradient algorithms. These methods all converge to the correct result. Since the LMS algorithm is not currently used in geophysical processing and we then look at how this method works on the deconvolution problem.

AN ELECTRICAL ENGINEERING PROBEM

Consider a sinusoid sampled N times per cycle and which is input sample by sample into an electrical system and then delayed by n samples, as shown in Figure 1. Both signals are recorded and are then weighted, summed and subtracted from a desired sinusoidal output to produce an error ε_k. Figure 1 shows the k^{th} sample of both the input sinusoid and desired sinusoid entering the system. This is a typical electrical engineering systems problem and is called an adaptive transversal filter (Widrow and Stearns, 1985).

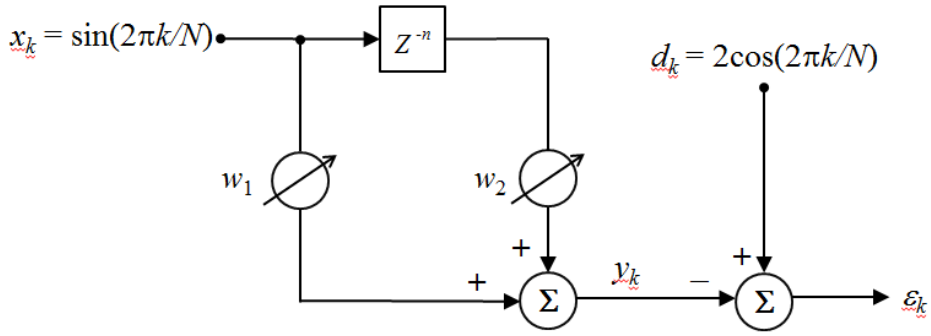

Figure 1: The problem as an electrical system, where an input sinusoid recorded and then shifted by n samples, and then the two signals are weighted and summed and subtracted from a desired signal to produce an error.

The same problem is shown in Figure 2 but now the two sinusoids (still with the same frequency and the second input shifted by n samples) are input as separate signals into two different channels. On output the signals are still weighted and summed and then subtracted from a desired signal to produce an error e_k.

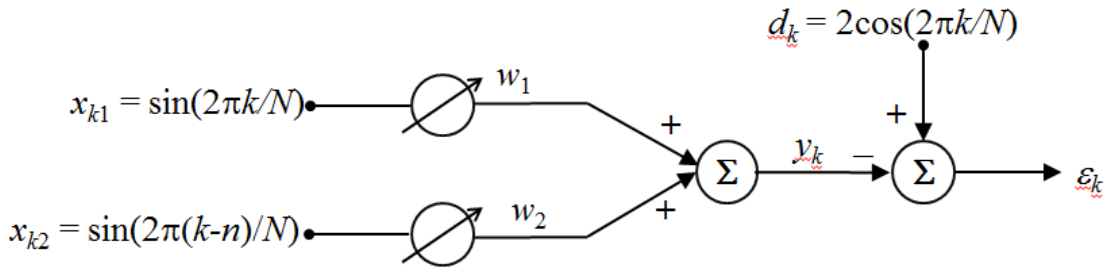

Figure 2: The same problem as in Figure 1, except that the two signals are recorded as separate channels, and then are weighted and summed and subtracted from a desired signal to produce an error.

Although the signals in Figure 2 are the same as those in Figure 1, there is no reason to assume that the two signals are related in any way, and we do this only to make the problem tractable. Thus, this way of conceptualizing the problem is much more amenable to seismic filtering and neural networks and is the approach we will adopt in this study.

Mathematically, we can write the problem in the following form:

$$y_k = w_1 x_{k1} + w_2 x_{k2} = d_k + \varepsilon_k \tag{1}$$

by explicitly writing the input terms, we can re-write equation 1 as:

$$w_1 \sin\left(\frac{2\pi k}{N}\right) + w_2 \sin\left(\frac{2\pi(k-n)}{N}\right) = 2\cos\left(\frac{2\pi k}{N}\right) + \varepsilon_k \tag{2}$$

We know that this problem has an exact solution from the trigonometric double angle formula that will always produce zero error, which you will recall is given by:

$$A\sin(\alpha-\beta) = A\sin(\alpha)\cos(\beta) - A\cos(\alpha)\sin(\beta).$$

Substituting the terms in equation 2 into the well known formula given above allows us to re-write the problem as:

$$w_2 \sin\left(\frac{2\pi k}{N} - \frac{2\pi n}{N}\right) = w_2 \sin\left(\frac{2\pi k}{N}\right)\cos\left(\frac{2\pi n}{N}\right) - w_2 \cos\left(\frac{2\pi k}{N}\right)\sin\left(\frac{2\pi n}{N}\right), \tag{3}$$

which leads to the following explicit solution for the weights:

$$w_1 = 2\frac{\cos\left(\frac{2\pi n}{N}\right)}{\sin\left(\frac{2\pi n}{N}\right)} = 2\cot\left(\frac{2\pi n}{N}\right) \text{ and } w_2 = -2\frac{1}{\sin\left(\frac{2\pi n}{N}\right)} = -2\csc\left(\frac{2\pi n}{N}\right). \tag{4}$$

Note that the above analysis assumes that we know the actual mathematical form of the input and desired signals, in which case the weights in Figure 1 and 2 could be pre-set to these values. However, the weights shown in both figures are adjustable and the key to the LMS algorithm is to adjust these weights after each set of values is received (i.e. the k^{th} set of inputs and desired signal). Later in the study we will get back to the LMS algorithm and how the weights are adjusted. However, first we will look at several other ways of computing the weights: the full least-squares method, the Newton method, and the steepest descent and conjugate gradient methods.

In geophysical measurements we record all the data first as a complete time series, rather than analyzing one sample at a time. So let us assume that in Figures 1 and 2 we run the system until we have recorded one full waveform of the two inputs and the

desired signal. If we let $N = 6$ and $n = 1$, so that the frequency and phase increments are both equal to $\pi/3$, then one complete period of the inputs and output requires that $k = 0, 1,\ldots,5$, and we can write equation 1 as the weighted sum of two vectors as shown here (where the error is now equal to zero):

$$y = 1.155\begin{bmatrix} 0 \\ 0.866 \\ 0.866 \\ 0 \\ -0.866 \\ -0.866 \end{bmatrix} - 2.309\begin{bmatrix} -866 \\ 0 \\ 0.866 \\ 0.866 \\ 0 \\ -0.866 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ -2 \\ -2 \\ -1 \\ 1 \end{bmatrix}.$$

Note that the numbers are repetitive since $\sin\left(\frac{\pi}{3}\right) = \frac{\sqrt{3}}{2} = 0.866$ and $\cos\left(\frac{\pi}{3}\right) = \frac{1}{2}$. Also, the weights shown above were computed analytically from the previous formulas and are equal to $w_1 = 1.155$ and $w_2 = -2.309$. In the more general N sample case with arbitrary inputs and output, the error is not zero and we can write:

$$\varepsilon = d - y = d - XW, \tag{5}$$

where $d = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{N-1} \end{bmatrix}, X = \begin{bmatrix} x_{01} & x_{02} \\ x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N-1,1} & x_{N-1,2} \end{bmatrix}$ and $W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$.

In the above notation, ε, d and y are now N length vectors, X is a N x 2 matrix that contains the input vectors as columns and W is a weight vector of length 2. Obviously, the above situation can be generalized to M input attributes, but using two inputs will allow us to show the results using two-dimensional plots.

In our case, where $N = 6$, we can write the result explicitly as:

$$\varepsilon = XW - d = \begin{bmatrix} 0 & -0.866 \\ 0.866 & 0 \\ 0.866 & 0.866 \\ 0 & 0.866 \\ -0.866 & 0 \\ -0.866 & -0.866 \end{bmatrix}\begin{bmatrix} 1.155 \\ -2.309 \end{bmatrix} - \begin{bmatrix} 0.5 \\ -1.5 \\ 3 \\ -3 \\ 1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

THE QUADRATIC ERROR SURFACE

Now, let us look at ways to compute the weights without knowing the explicit form of the inputs. Squaring the error using vector notation, we get:

$$\varepsilon^T \varepsilon = d^T d + W^T X^T X W - 2 d^T X W \,. \tag{6}$$

Equation 6 can be re-written in the following way:

$$\sum_{k=1}^{N} \varepsilon_k^2 = \sum_{k=1}^{N} d_k^2 - W^T R W - 2 P^T W \,, \tag{7}$$

where $R = X^T X$ and $P^T = d^T X$. The matrix R is the cross-correlation matrix of the input vectors and the vector P is the cross-correlation of the desired output with the input. Note that to make these matrices true covariance matrices they should be normalized by dividing by N. However, in geophysical analysis this is often not done and in our example the values are easier to interpret if they are not normalized by dividing by 6. In our numerical example, R is computed as:

$$R = \begin{bmatrix} 0 & 0.866 & 0.866 & 0 & -0.866 & 0.866 \\ -0.866 & 0 & 0.866 & 0.866 & 0 & -0.866 \end{bmatrix} \begin{bmatrix} 0 & -0.866 \\ 0.866 & 0 \\ 0.866 & 0.866 \\ 0 & 0.866 \\ -0.866 & 0 \\ -0.866 & -0.866 \end{bmatrix} = \begin{bmatrix} 3 & 1.5 \\ 1.5 & 3 \end{bmatrix},$$

and P as:

$$P = \begin{bmatrix} 0 & 0.866 & 0.866 & 0 & -0.866 & 0.866 \\ -0.866 & 0 & 0.866 & 0.866 & 0 & -0.866 \end{bmatrix} \begin{bmatrix} 0.5 \\ -1.5 \\ 3 \\ -3 \\ 1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ -5.196 \end{bmatrix}$$

Before discussing the many ways that we can solve for the weights, note that the difference between the squared sum of the errors and the squared sum of the desired values results in a single value that we can call the quadratic error (*QE*):

$$QE = \sum_{k=1}^{N} \varepsilon_k^2 - \sum_{k=1}^{N} d_k^2 = W^T R W - 2 P^T W \,. \tag{8}$$

This suggests a "trial-and-error" approach to finding the correct weights by computing the quadratic error for a wide range of weight values and identifying the minimum value of the error. This involves searching the quadratic error surface, as shown in Figure 3.

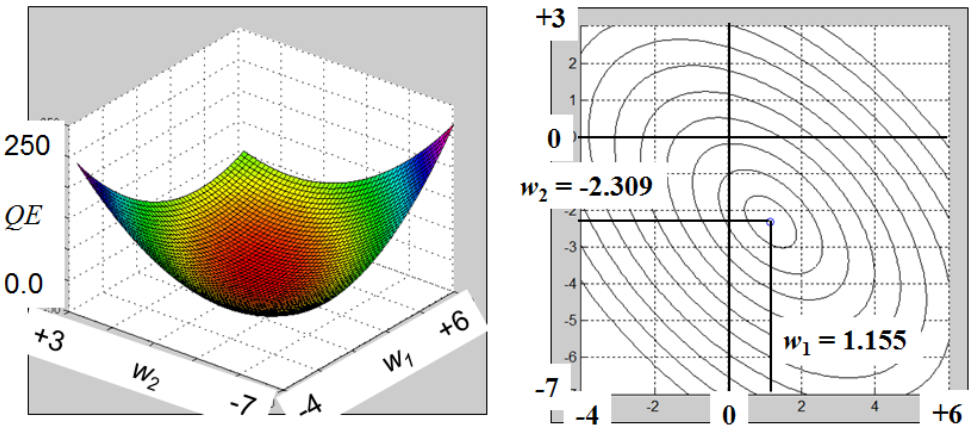

Figure 2: The quadratic error surface from equation 8, where on the left is a 3D view of the error surface and on the right is the contoured map, where we note that the correct weights are found at the minimum error.

As seen in Figure 2, the minimum of the error surface, which looks like an elliptical bowl, is indeed at the correct result. In the next section we will look at ways of making an estimate of the weights and then updating this estimate and moving down the "bowl" until we reach the minimum.

THE LEAST-SQUARES SOLUTION

To move from one point on the bowl in Figure 2 to the minimum, we first need to compute the gradient. The gradient of the mean-square-error can be obtained by differentiating the error with respect to the weights, or:

$$\nabla = \frac{\partial \xi}{\partial W} = RW - P, \tag{9}$$

where $\xi = \sum_{k=1}^{N} \varepsilon_k^2$ is the squared error. If we had normalized the covariances we would have also divided this term by N and it would be the mean-square-error, but again we will use the un-normalized terms. Also note that theoretically there should be a factor two in front of both R and P in equation 9. Many authors, including Widrow and Stearns (1985), include this factor in their derivations. However, the discussion is identical, and a bit simpler, if we remove the factor of 2, as do most books on optimization (e.g. Murray et al., 1986). The obvious solution to equation 9 is to set the gradient to zero and invert R, giving:

$$W = R^{-1}P. \tag{10}$$

Equation 10 is called the least-squares solution, and in our case gives the correct answer, as shown below:

$$W = \begin{bmatrix} 3 & 1.5 \\ 1.5 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -5.196 \end{bmatrix} = \begin{bmatrix} 0.444 & -0.222 \\ -0.222 & 0.444 \end{bmatrix} \begin{bmatrix} 0 \\ -5.196 \end{bmatrix} = \begin{bmatrix} 1.155 \\ -2.309 \end{bmatrix}.$$

But for large geophysical and neural network problems, the least-squares solution is rarely an option due to the size of the datasets, and we need to find more efficient methods which involving "searching" for a solution. These methods will be discussed in the next section.

GRADIENT DESCENT METHODS

All search methods for finding the best solution for a linear problem (i.e. the minumum shown in Figure 2 start with an initial guess set of weights and then iterate towards the answer using the gradient computed in equation 9 in some way. That is:

$$W_{i+1} = W_i + \Delta W_i, \text{ where } i = 0,1,\ldots,M\,. \tag{11}$$

If you know the correlation matrix R, a technique called Newton's method will find the solution in a single iteration, or

$$W_1 = W_0 - R^{-1}\nabla_0\,. \tag{12}$$

Substituting for the gradient makes it clear why this is so:

$$W_1 = W_0 - R^{-1}\left(RW_0 - P\right) = W_0 - R^{-1}RW_0 + R^{-1}P\,.$$

Simplifying, we get the result

$$W_1 = \left(W_0 - W_0\right) + R^{-1}P = R^{-1}P\,.$$

Figure 4 shows that regardless of our initial guess, we converge to the correct answer in one step using Newton's method.

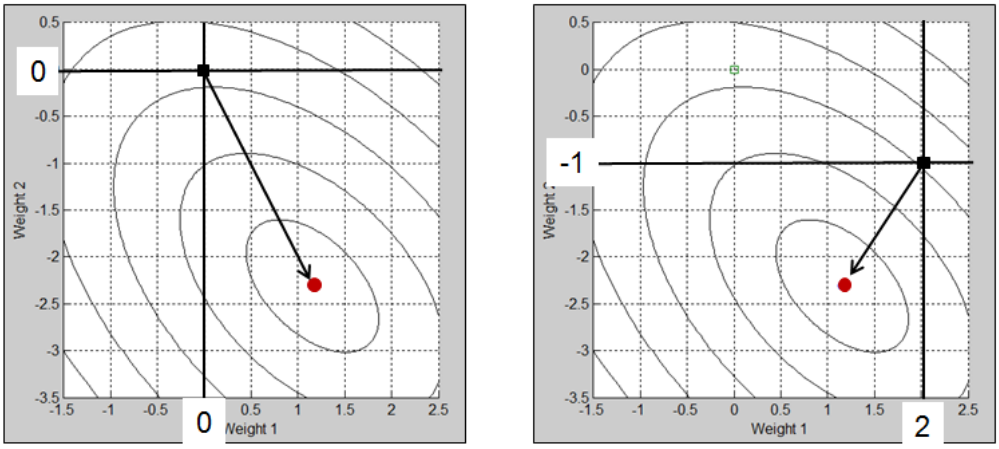

Figure 4: In Newton's method, given in equation 12, we converge to the correct solution in one iteration no matter whether our starting guess is (0,0) as shown on the left or (2,-1) as shown on the right.

But Newton's method is a bit of a "cheat", since if we have to compute the matrix inverse, why not just perform the full inversion? The most common iterative method in which we do not compute the full inverse first is called steepest descent, or gradient descent, and can be written very simply as:

$$W_{i+1} = W_i - \alpha_i \nabla_i, \; i = 0, \ldots, M, \tag{13}$$

where α is called the step size. The most crucial choice in steepest descent is therefore the value of the step size α. It can be shown the value of the step size should be somewhere between 0 and the 2 divided by the largest eigenvalue of the matrix R, or:

$$0 < \alpha < \frac{2}{\lambda_{\max}}, \text{ where } \lambda_{\max} = \text{largest eigenvalue of } R.$$

The eigenvalues of R are λ_{max} = 4.5 and λ_{min} = 1.5, so $2/\lambda_{max}$ = 0.444 is the largest value before the steepest descent algorithm will become unstable. The optimum value of α is given by the equation:

$$\alpha_i = \frac{\nabla_i^T \nabla_i}{\nabla_i^T R \nabla_i} = 0.333 \text{ for the first iteration. .} \tag{14}$$

Using equation 14, the step size is thus re-computed after each iteration of the steepest descent algorithm. This value equals 0.333 for the first step, which is the average of the two eigenvalues (there is probably a mathematical reason for this, but the author has not figured it out!). Equation 14 is called line minimization and is the optimal solution in most cases. However, in some problems this can lead to instability and you are better off just fixing a constant value for the step size using trial and error. A small step size will result in more iterations, but the path will be very smooth and gradual. A large step size will converge faster and will thus require fewer iterations, but it runs the risk of causing instability.

Figure 5 shows the path taken by the steepest descent algorithm using the optimum step size given in equation 14, starting at an initial guess of $W_0^T = [0, 0]$.

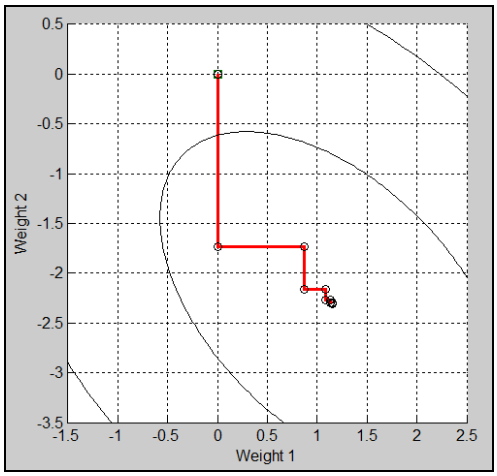

Figure 5: The steps in the steepest descent algorithm for our example.

Note also in Figure 5 that each step is orthogonal to the previous step. Finally, note that the first step is the largest and for a step size of 0.333 it is computed as:

$$W_1 = W_0 - \alpha_0 \nabla_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 0 \\ 5.2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.733 \end{bmatrix}. \tag{15}$$

The conjugate gradient (CG) algorithm (Hestenes and Steifel, 1952) is a refinement of steepest descent. The CG method is not discussed by Widrow and Stearns (1985) since it falls more into the domain of mathematical optimization (Gill et al., 1986). It starts with the first step from steepest descent, as defined in equation 15. Next, the algorithm finds the "conjugate" to this first direction, where conjugate vectors p_i and p_j are defined in the following way using our correlation matrix R:

$$p_i^T R p_j = 0, \text{ where } i \neq j. \tag{16}$$

Without going into the details (see Hagan et al., 1996 for an excellent derivation of the algorithm), the conjugate gradient algorithm can be written:

$$W_{i+1} = W_i + \alpha_i p_i, \tag{17}$$

where $p_0 = -\nabla_0$ and $p_i = -\nabla_i + \beta_i p_{i-1}$, where $\beta_i = \dfrac{\nabla_i^T \nabla_i}{\nabla_{i-1}^T \nabla_{i-1}}$. Note that each step utilizes only the gradient calculation but does so in a way to ensure the conjugacy of each successive step. To show that this works on our dataset, note that the calculations for the first two conjugate gradient directions give:

$$p_0 = -\nabla_0 = \begin{bmatrix} 0 \\ -5.2 \end{bmatrix}, p_1 = \begin{bmatrix} 2.6 \\ -1.3 \end{bmatrix} \Rightarrow p_0^T R p_1 = \begin{bmatrix} 0 & -5.2 \end{bmatrix}\begin{bmatrix} 3 & 1.5 \\ 1.5 & 3 \end{bmatrix}\begin{bmatrix} 2.6 \\ -1.3 \end{bmatrix} = 0,$$

The convergence of conjugate gradient algorithm when applied to our problem is shown in Figure 6 as the red curve.

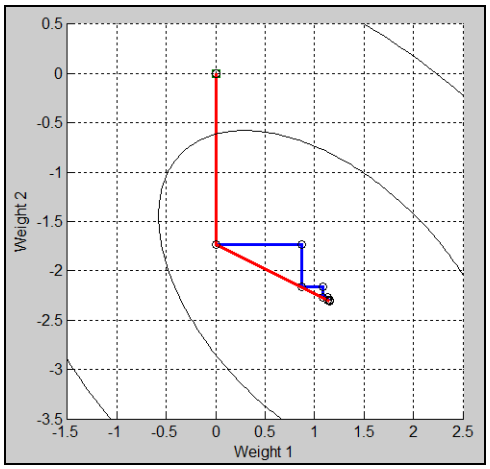

Figure 6: The steps in the conjugate gradient (CG) algorithm for our example (red). Note that CG converges in two steps, which is the number of weights being in the problem. The steepest descent algorithm is shown in blue.

For comparison purposes, the steepest descent path is shown in blue in Figure 6. Note that the conjugate gradient method shown in Figure 6 converges to the correct answer in two steps. It can be shown mathematically that the conjugate gradient method always converges to the correct answer in the same number of iterations as the number of unknown weights. Note that the first step for both steepest descent and conjugate gradient is identical. The second step of the conjugate gradient method gives the correct weights, as shown in the following calculation:

$$W_2 = W_1 + \alpha_1 p_1 = \begin{bmatrix} 0 \\ -1.733 \end{bmatrix} + 0.444 \begin{bmatrix} 2.6 \\ -1.3 \end{bmatrix} = \begin{bmatrix} 1.15 \\ -2.31 \end{bmatrix}.$$

Note in the above calculation that α_1 equals the maximum allowable step size of 2 divided by the largest eigenvalue. (Again, there must be a good theoretical explanation for this, but the author has not found it!).

THE LEAST-MEAN-SQUARE (LMS) ALGORITHM

Finally, we shall discuss the Least-Mean-Square, or LMS, algorithm. Note that the gradient descent methods we described in the last section, steepest descent and conjugate gradient, assumed that all the data had been recorded before the analysis started, and therefore the correlation matrix and cross-correlation vector could be calculated. However, what if the data arrives sample by sample, as in our initial example? In this case remember that the error at each input sample is given by:

$$\varepsilon_k = d_k - y_k = d_k - X_k W_k = d_k - \begin{bmatrix} x_{k1} & x_{k2} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_k . \tag{18}$$

Thus, we are now going back to the situation shown in Figures 1 and 2.

Also recall the definition of the gradient, and note that for our two weight example we can write:

$$\nabla_k = \begin{bmatrix} \dfrac{\partial \varepsilon_k^2}{\partial w_1} \\ \dfrac{\partial \varepsilon_k^2}{\partial w_2} \end{bmatrix} = 2\varepsilon_k \begin{bmatrix} \dfrac{\partial \varepsilon_k}{\partial w_1} \\ \dfrac{\partial \varepsilon_k}{\partial w_2} \end{bmatrix} = -2\varepsilon_k \begin{bmatrix} x_{k1} \\ x_{k2} \end{bmatrix} = -2\varepsilon_k X_k^T . \tag{19}$$

This suggests that we can update the weights after each sample, as follows:

$$W_{k+1} = W_k - \alpha \nabla_k = W_k + 2\alpha \varepsilon_k X_k^T . \tag{20}$$

This is called the Least-Mean-Square, or LMS, algorithm. Since the full correlation matrix is not available, the step size α cannot be calculated for each step, so it is set to a reasonable value for the complete set of iterations. We can now update Figure 2 to show the LMS algorithm in block diagram form. This diagram is shown in Figure 7. Note that this is a feedback approach in which the weights are adaptively updated.

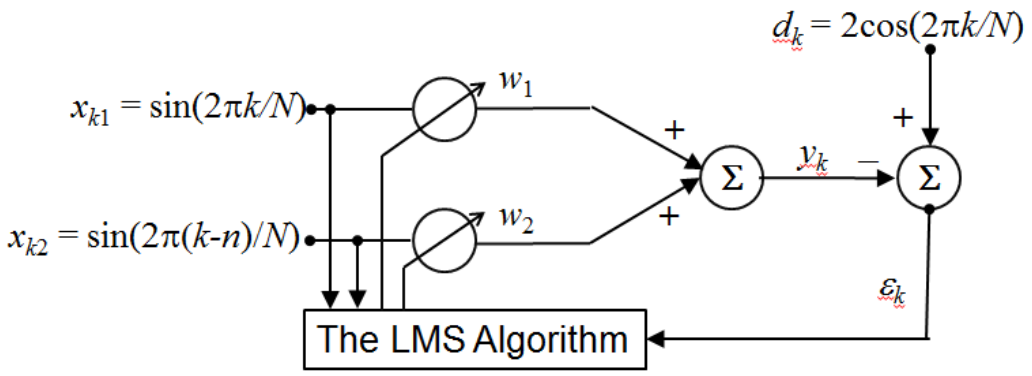

Figure 7: In the LMS algorithm the error is fed back into the algorithm after each sample is input and the weights are adjusted.

The LMS solution to our problem is shown in Figure 8, with α set to 0.333, as in the first step of steepest descent and conjugate gradient. In Figure 8, the path taken by the LMS algorithm is shown in red. The conjugate gradient path is shown in blue for comparison purposes. Note that the LMS path is chaotic at first but soon takes on orthogonal steps like steepest descent.

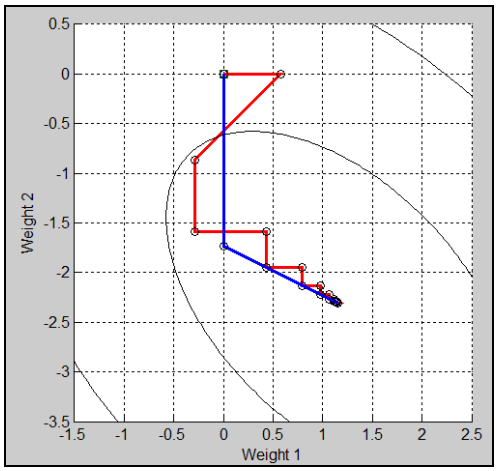

Figure 8: The steps in the LMS algorithm for our example are shown in red. The steps in the conjugate gradient algorithm are shown in blue.

Figure 9 shows the convergence of the LMS algorithm. In the figure, the red curve shows the true desired sinusoid and the blue curve shows the LMS prediction after each iteration. This plot is interesting for two reasons. First, in our previous discussion of the full least-squares inverse, Newton's algorithm, and the steepest descent and conjugate gradient algorithms, note that we only used one period of the sinusoid, or 6 samples. In Figure 8 we see that after only 6 iterations, the LMS algorithm has got the shape of the curve fairly correct, but not its amplitude. Since we are dealing with simple sinusoids in this problem, we can continue to feed values into the LMS algorithm. Note that it the fit is not perfect until after about 20 iterations.

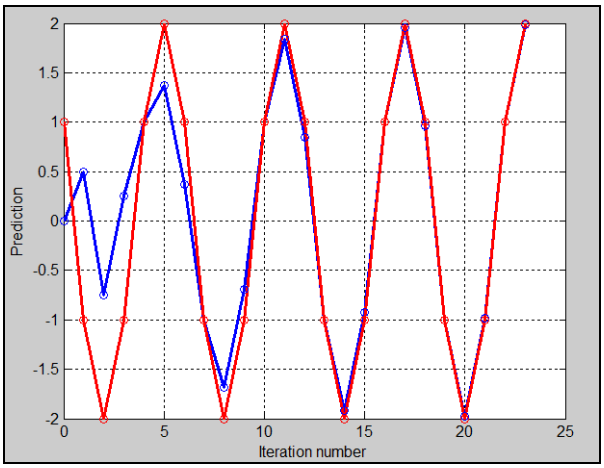

Figure 9: Convergence to the correct answer using the LMS algoithm as a function of iteration number, where the red curve is the correct answer and the blue curve is the LMS result. Note that it has converged after 20 iterations.

A DECONVOLUTION EXAMPLE

In this last section, we will apply the theory just developed for a sinusoidal example to a geophysical example. Specifically, we will look at wavelet deconvolution (see, for example, Claerbout, 1976). This starts with and understanding of the forward convolutional model shown in Figure 10. In this figure, we convolve the simple zero-phase wavelet shown at the top of the figure with the two-spike reflectivity shown in the middle panel to produce the output seismic trace at the bottom of the figure.

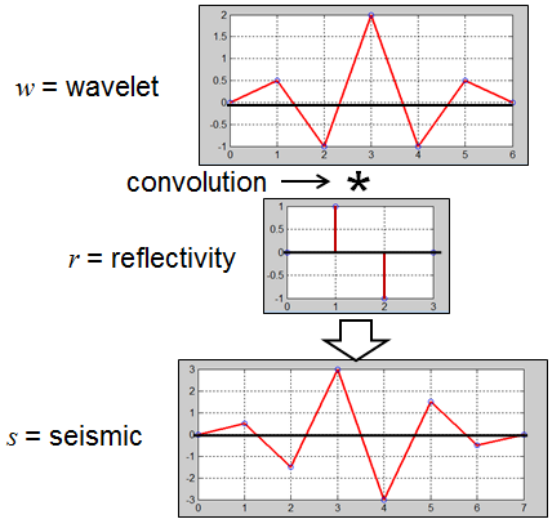

Figure 10: The wavelet in the top panel is convolved with the reflectivity in the middle panel to produce the seismic trace in the bottom panel.

The wavelet can be written in vector format as $w^T = [0.5 \quad -1 \quad 2 \quad -1 \quad 0.5]$, where we have dropped the zeros at each end of the wavelet shown in Figure 9. The reflectivity can be written as $r^T = [1 \quad -1]$, where again we have dropped the zeros, and the seismic trace

as $s^T = \begin{bmatrix} 0.5 & -1.5 & 3 & -3 & 1.5 & -0.5 \end{bmatrix}$. There are many different ways of writing the convolution operation, but the easiest to visualize is the convolutional matrix approach, in which the wavelet appears as shifted columns in a wavelet matrix, where there are as many columns as there are reflection coefficients, and this matrix is multiplied times the reflectivity vector. This approach is shown below:

$$Wr = s\,, \tag{21}$$

where $W = \begin{bmatrix} 0.5 & 0 \\ -1 & 0.5 \\ 2 & -1 \\ -1 & 2 \\ 0.5 & -1 \\ 0 & 0 \end{bmatrix}$ and $r = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Rightarrow s = \begin{bmatrix} 0.5 & 0 \\ -1 & 0.5 \\ 2 & -1 \\ -1 & 2 \\ 0.5 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1.5 \\ 3 \\ -3 \\ 1.5 \\ -0.5 \end{bmatrix}$.

Note that this is a classic "thin-bed tuning" example, since the final seismic trace, which is the convolution of a zero-phase wavelet with a closely spaced dipole, looks like a single ninety degree phase wavelet. This is a very difficult problem to solve, because in general we do not know the true wavelet shape. However, in this problem we will assume we know the correct shape of the wavelet as well as the observed seismic trace. and can solve the problem using the least-squares method:

$$r = R^{-1}P, \text{ where } R = W^T W \text{ and } P = W^T s\,. \tag{22}$$

Numerically, R and P can be computed as:

$$R = \begin{bmatrix} 0.5 & -1 & 2 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 2 & -1 & 0.5 \end{bmatrix} \begin{bmatrix} 0.5 & 0 \\ -1 & 0.5 \\ 2 & -1 \\ -1 & 2 \\ 0.5 & -1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 6.5 & -5 \\ -5 & 6.5 \end{bmatrix},$$

$$\text{and } P = \begin{bmatrix} 0.5 & -1 & 2 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 2 & -1 & 0.5 \end{bmatrix} \begin{bmatrix} 0.5 \\ -1.5 \\ 3 \\ -3 \\ 1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 11.5 \\ -11.5 \end{bmatrix},$$

which gives the correct answer as shown below:

$$r = R^{-1}P = \begin{bmatrix} 0.377 & 0.29 \\ 0.29 & 0.377 \end{bmatrix} \begin{bmatrix} 11.5 \\ -11.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Next, let's see how well our gradient search methods perform. First, we will look at the quadratic error plot, shown in Figure 11, in which we have indicated the correct answer as the red circle and an initial guess of $r^T = [-1, -1]$.

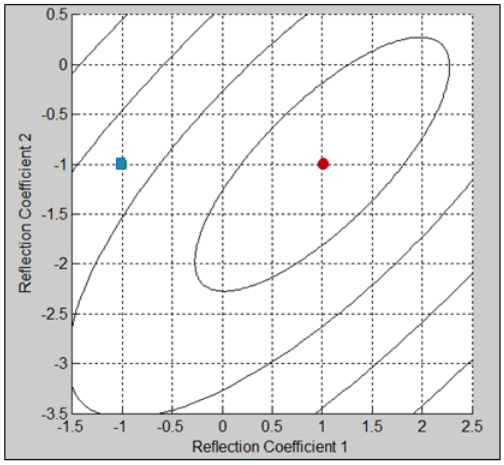

Figure 11: The quadratic error plot for the deconvolution example, where the red circle is the correct answer and the blue square is the starting guess for the gradient search methods.

The steepest descent path is then shown in Figure 12. Note that steepest descent has converged to the correct answer and has actually converged quite rapidly, in about four steps.

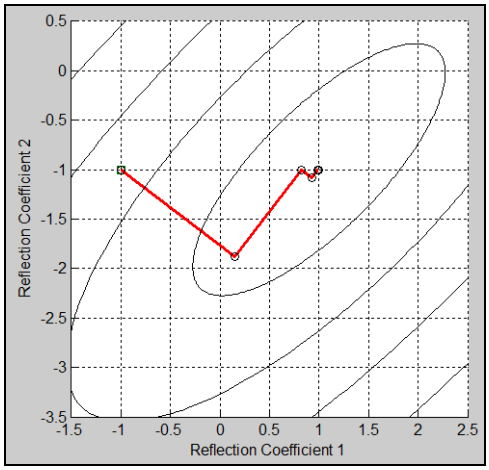

Figure 12: The path for the steepest descent algorithm, which has converged to the correct result.

The conjugate gradient solution is shown in Figure 13 and, as expected, has converged in two iterations.

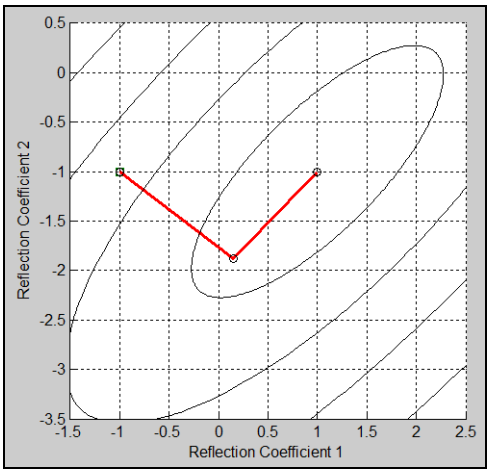

Figure 13: The path for the conjugate gradient algorithm, which has converged to the correct result.

In creating the results shown in Figures 12 and 13, several interesting differences between steepest descent and conjugate gradient came to light. It turns out that in using the line minimization method of equations 14 and 17 (for the α and β terms) both methods "blow up" after the first iteration for some choices of the initial guess. (For example, an initial guess of r^T = [0, 0] causes this problem, which is why I did not use this as a guess). This problem is easily fixed in steepest descent by using the simpler approach of finding a constant step size by trial and error. However, the conjugate gradient method is based on finding these optimal parameters, and thus does not give any added advantage over steepest descent in the case of instability.

Finally, Figure 14 shows the result of the least-mean-square, or LMS, algorithm. In this case the path is moving in the right direction but stops at values of r^T = [0.5, -1] before converging to the correct result. This is because of the few samples presented to it in this simpler problem. In the more realistic problem where the seismic trace and reflectivity are much longer, the algorithm should converge correctly. In fact, there is the added advantage that since LMS is an adaptive algorithm, a time varying wavelet could be estimated.

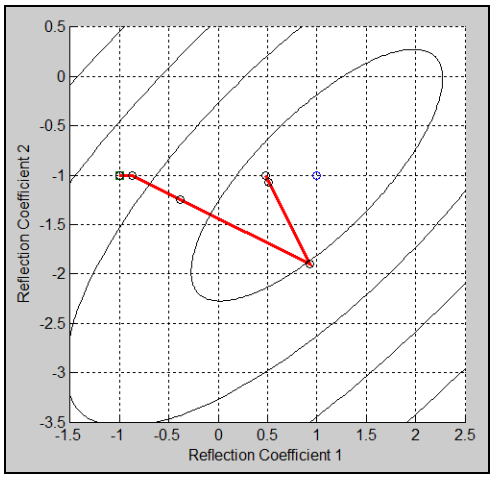

Figure 14: The path for the LMS algorithm, which has not converged to the correct result due to an insufficient number of measurements.

CONCLUSIONS

In this report, I have presented a tutorial on linear optimization techniques, such as the steepest descent and gradient descent algorithms, but have included an algorithm that is never discussed in the mathematical literature on optimization, the least-mean-square, or LMS, algorithm. The LMS algorithm is an adaptive algorithm that was developed in the electrical engineering community and has found great success in such applications as the adaptive equalization of telephone channels and blood pressure regulation (see Widrow and Stearns, 1985).

The key example used in this talk involved computing the weights which, when applied to two sampled sinusoids of the same frequency but different phase, produced an output which would cancel a third sinusoid of the same frequency and a different phase and amplitude. This example was used for two reasons. First, the weights can be computed exactly from a trigonometric identity, so we know what the right answer is from the start. Second, because a sinusoid is infinitely repeating, cycle by cycle, we can make the input and desired output as long or as short as we want. Thus, for the least-squares, steepest descent and conjugate gradient algorithms, we used only the first cycle of our sampled sinusoids, which involved only 6 samples. For the LMS algorithm, which adapts on a sample by sample basis, we were able to continue the input and output for three cycles, until the algorithm converged.

In geophysics, we make great use of both the steepest descent and conjugate gradient algorithms since we always have our data recorded before we analyze it. Thus, our second example showed how to solve the deconvolution problem using least-squares, steepest descent and conjugate gradient. However, the LMS algorithm is rarely used in geophysical processing and we therfore looked at how this method would work on a deconvolution problem. Although the method did not converge completely in our example because of the lack of samples, it would work well on longer datasets and also potentially solve the problem of time-varying wavelets.

The one area that has made use of all three methods, particularly the LMS algorithm, is neural network analysis and machine learning (Rummmelhart et al., 1986), which we

did not discuss today. Neural networks are being used more and more in geophysical analysis, so an understanding of the LMS algorithm will lead naturally to an understanding of neural networks.

REFERENCES

Claerbout, J, 1976, Fundamentals of Geophysical Data Processing: McGraw Hill, Inc.

Gill, P.E., Murray, W., and Wright, M.H., 1981, Practical Optimization: Academic Press, New York.

Hagan, M.T., Demuth, H.B., and Beale, M., 1996, Neural Network Design: PWS, Boston.

Hestenes, M.R., and Stiefel, E., 1952, Methods of conjugate gradients for solving linear systems: Journal of Research of the National Bureau of Standards, 29, 409-439.

Rosenblatt, M., 1958, The perceptron: A probabilistic model for information storage and organization in the brain: Psychological Review, 65, 386-408.

Rummelhart, D.E., Hinton, G.E., and Williams, R.J., 1986, Learning representations of back-propagation errors: Nature, 323, 533-536.

Widrow, B., and Hoff, M.E., 1960, Adaptive switching circuits: *IRE WESCON* Conv. Rec., pt. 4, p 96-104.

Widrow, B., and Stearns, S., 1985, Adaptive Signal Processing: Prentice-Hall, New York.

ACKNOWLEDGEMENTS

I want to thank my colleagues at the CREWES Project and at CGG and Hampson-Russell for their support and ideas, as well as the sponsors of the CREWES Project.