11-1-1993

Exact and Approximate Methods for Computing the Hessian of a Feedforward Artificial Neural Network

Craig W. Codrington
Purdue University School of Electrical Engineering

Manoel F. Tenorio
Purdue University School of Electrical Engineering

ADVENTURES IN CUBIC CURVE FITTING: TWO OPTIMIZATION TECHNIQUES AND THEIR APPLICATION TO THE TRAINING OF NEURAL NETWORKS

CRAIG W. CODRINGTON
ANTONIO G. THOME
MANOEL F. TENORIO

TR-EE 93-42
NOVEMBER 1993

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Adventures in Cubic Curve Fitting: Two Optimization Techniques and their Application to the Training of Neural Networks

Craig W. Codrington Antonio G. Thome

Manoel F. Tenorio

Abstract

We present two optimization techniques based on cubic curve fitting; one based on function values and derivatives at two previous points, the other based on derivatives at three previous points. The latter approach is viewed from a derivative space perspective, obviating the need to compute the vertical translation of the cubic, thus simplifying the fitting problem. We demonstrate the effectiveness of the second method in training neural networks on parity problems of various sizes, and compare our results to a modified Quickprop algorithm and to gradient descent.

1 Introduction

Gradient descent, in the form of the well-known backpropagation algorithm, is frequently used to train feedforward neural networks, i.e. to find the weights which minimize some error measure f. However, it's slow rate of convergence has prompted investigations into second order methods, such as Newton's method:

$$\vec{w}_{k+1} = \vec{w}_k - H(\vec{w}_k)^{-1} \nabla f(\vec{w}_k) \tag{1}$$

where $H(\vec{w}_k)$ is the Hessian matrix. To avoid the computationally expensive task of computing the exact Hessian, quasi-Newton type approximations are often made, based on past gradient and weight differences. Such methods can be viewed as minimizing a quadratic surface q fitted to a set K of previous points, such that the gradient of the quadratic approximates the gradient of f at each point, i.e.

$$\nabla q(\vec{w}_k) \approx \nabla f(\vec{w}_k) \quad \text{for each } k \in K \tag{2}$$

However, the same quadratic may not hold over the entire sequence of n + 1 points required to estimate the Hessian, particularly when there is no local

minimum nearby, or when the steps are large. Alternatively, by imposing a structure on the Hessian (diagonal, for example), one can use fewer points, but there is no guarantee that the resulting estimate will be close to the true Hessian. In this paper we go one step further and treat the weights as being entirely decoupled (although in reality they are not), thus reducing the problem to a series of 1D optimizations along each weight axis. The resulting reduction in complexity allows a higher order curve to be fit, which would theoretically mean a higher rate of convergence, were it not for coupling effects between the weights. It also means that the fitting can be based on fewer points, allowing more rapid adaptation to the local characteristics of f.

An independent "greedy" 1D minimization along each weight axis yields a solution that is less optimal than the cooperative solution, which takes account of coupling. It may be advantageous, however, when the true Hessian is not positive definite; many quasi-Newton type methods construct a positive definite approximation to the Hessian, which can introduce a somewhat arbitrary perturbation of the Newton step (1) which acts along all weight axes. Minimizing along each weight axis independently decouples these perturbations, so that the adjustment resulting from a well-behaved fit in one dimension is not altered by the perturbation which must be introduced to

control a poorly behaved fit in another.

This strategy proved very effective in the Quickprop algorithm [1, 2], which uses the Method of False Position [3, pages 202–2031 to fit a quadratic to f along each weight dimension independently. The component of the next point $\vec{w}_{k+1}$ corresponding to each weight w is chosen to minimize the quadratic:

$$w_{k+1} = w_k - \frac{f'_k}{f'_k - f'_{k-1}}(w_k - w_{k-1}) \tag{3}$$

where $f'_k = f'(w_k)$ and $f'_{k-1} = f'(w_{k-1})$ are the respective components of $\nabla f(\vec{w}_k)$ and $\nabla f(\vec{w}_k)$ corresponding to w. Quickprop also adds a few heuristics to improve convergence and enhance stability; one is to bound the rate of increase of $|w_{k+1} - w_k|$ by an empirically tuned parameter called the maximum growth rate (we will return to this later).

The Method of False Position **(3)** can be derived either by fitting a quadratic to one function value (f_k) and two derivatives $(f'_k$ and $f'_{k-1})$, or by approximating the second derivative in Newton's Method (1) by

$$f''(w_k) \approx \frac{f'_k - f'_{k-1}}{w_k - w_{k-1}} \tag{4}$$

Since the function value f_k does not appear in the Quickprop update

equation (3), it can be interpreted as finding the zero of a line fitted to two points $((w_k, f'_k)$ and $(w_{k-1}, f'_{k-1}))$ in "derivative space" (fig 1b), as well as minimizing a quadratic fitted in "function space" (fig 1a). In one of the algorithms we have developed, CubicpropII, this concept will be extended by fitting a quadratic to three points in derivative space, which is the same as fitting a cubic in function space. Intuitively, a derivative space interpretation is possible when knowledge of only one function value (e.g. f_k) is assumed, since this can only constrain the "vertical" translation of the fitted (function space) curve, which is of no consequence if we are searching for a local minimum.

2 CubicpropI

Inspired by Quickprop, which fits a quadratic to one function value (f_k) and two derivatives (f'_k, and f'_{k-1}), we attempted to fit a cubic to two function values (f_k and f_{k-1}) and two derivatives (f'_k, and f'_{k-1}). The next point w_{k+1} is chosen as the local minimum of the fitted cubic, resulting in the following update equation:

$$w_{k+1} - w_k = \frac{|w_k - w_{k-1}|u_2 - (w_k - w_{k-1})(f'_k + u_1)}{f'_k + f'_{k-1} + 2u_1} \tag{5}$$

where

$$u_1 = f'_k + f'_{k-1} - 3\frac{f_k - f_{k-1}}{w_k - w_{k-1}} \tag{6}$$

and

$$u_2 = \sqrt{u_1^2 - f'_k f'_{k-1}} \tag{7}$$

Luenberger [3, pages 205–206] states a similar result, but his formula is not correct, since it is not invariant under permutations of the current and previous points (this can be verified by applying his formula to a cubic; if for two given points w_k and w_{k-1} the formula gives the minimum, then interchanging the two points will give the maximum, and conversely).

The fitted cubic will have no local minimum when (7) is complex; in this case an alternative strategy, such as a gradient descent step, must be used. An alternative strategy must also be used when the adjustment indicated by (5) is not in the negative gradient direction; this occurs, for example, when $|f'_k| > |f'_{k-1}|$ and $f_k < f_{k-1}$ (figure 2).

Although (5) will find the local minimum of a cubic in one step, it was found to be ill-suited to training neural networks. On the flat plateaus which characterize the error surface of such networks, the algorithm tends to fit a cubic which appears flat in the vicinity of the current and previous points (figure 3a), but on closer examination is seen to have its minimum

and maximum near the current and previous points, respectively (figure 3b); the resulting step is frequently very small.

3 CubicpropII

Quickprop can be viewed as finding the minimum of a quadratic fitted in function space, or equivalently, as finding the zero of a line fitted to two points $((w_k, f'_k)$ and $(w_{k-1}, f'_{k-1}))$ in derivative space. The natural 'extension to this is to fit a quadratic to three points $((w_k, f'_k)$, (w_{k-1}, f'_{k-1}) and $(w_{k-2}, f'_{k-2}))$ in derivative space; the next point is then chosen to be the zero of the quadratic corresponding to the (local) minimum of the cubic in function space (figure 4). The resulting update equation, which we call cubicpropII, is given by:

$$w_{k+1} - w_k = \frac{u_1 - f'_k(w_k - w_{k-2})(w_{k-1} - w_{k-2}) + su_2}{f'_k(w_{k-1} - w_{k-2}) - f'_{k-1}(w_k - w_{k-2}) + f'_{k-2}(w_k - w_{k-1})} \quad (8)$$

where

$$u_1 = \frac{1}{2}(f'_k(w_{k-1} - w_{k-2})^2 + f'_{k-1}(w_k - w_{k-2})^2 - f'_{k-2}(w_k - w_{k-1})^2) \quad (9)$$

$$u_2 = \sqrt{u_1^2 - f'_k f'_{k-1}(w_k - w_{k-2})^2(w_{k-1} - w_{k-2})^2} \quad (10)$$

$$s = \text{sign}\left((w_k - w_{k-2})(w_k - w_{k-1})(w_{k-1} - w_{k-2})\right) \tag{11}$$

The above update equation can also be expessed as

$$w_{k+1} - w_k = \frac{u_1 - 2(f'_k - f'_{k-1})(w_k - w_{k-2})(w_{k-1} - w_{k-2}) + su_2}{2u_0} \tag{12}$$

where

$$u_0 = f'_k(w_{k-1} - w_{k-2}) - f'_{k-1}(w_k - w_{k-2}) + f'_{k-2}(w_k - w_{k-1}) \tag{13}$$

$$u_1 = f'_k(w_{k-1} - w_{k-2})^2 + f'_{k-1}(w_{k-1} - w_{k-2})(w_k - 2w_{k-1} + w_{k-2}) - f'_{k-2}(w_k - w_{k-1})^2) \tag{14}$$

$$u_2 = \sqrt{u_1^2 - 4f'_{k-1}u_0(w_k - w_{k-2})(w_k - w_{k-1})(w_{k-1} - w_{k-2})} \tag{15}$$

As in cubicpropI, an alternative strategy must be used when (10) is complex or when the adjustment indicated by (8) is not in the negative gradient direction relative to the current point; we initially tried using a gradient descent step instead of the cubic adjustment in these cases. In fact, gradient descent was used whenever (10) was complex or the first derivative of the fitted quadratic (i.e. the second derivative of the corresponding function

space cubic) was negative when evaluated at the current point, i.e.

$$2aw_k + b < 0 \tag{16}$$

where a and b are given in the Appendix. This is a more stringent condition which implies that the adjustment will be in the negative gradient direction relative to the current point.

The resulting algorithm provided good stability and rapid convergence, and was a clear improvement over cubicpropI. However, it suffered from a tendency to become trapped in local minima. To counteract this, we applied a perturbation in the form of a gradient descent term and a momentum term which were always added to the cubic adjustment (8). When the cubic adjustment could not be applied because (10) was complex or the first derivative of the quadratic was negative when evaluated at the current point, the gradient and momentum terms were used alone. This yielded a dramatic improvement in performance.

4 Results

The CubicpropII algorithm was used to train neural networks on parity problems of various sizes. The n-parity function maps each of the 2^n possible

n-bit binary input patterns with an even number of ones to 0, and those with an odd number to 1. However, since the hyperbolic tangent nonlinearity was used in the network, a -1/1 parity function was used instead of the 0/1 parity function described above. Performance was measured in terms of the Normalized Root Mean Squared Error (NRMSE), defined as

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{P}\sum_{p=1}^{P}(y_p - z_p)^2}}{\sqrt{\frac{1}{P}\sum_{p=1}^{P}(y_p - \bar{y})^2}} \tag{17}$$

where P is the number of patterns, $z_1, \ldots, z_P$ are the outputs of the network and $y_1, \ldots,$ yp are the desired outputs, with sample mean $\bar{y}$.

Table 1 shows the results for CubicpropII, Gradient Descent, and a, modified Quickprop algorithm on the parity-2 (XOR) problem, using a network with 2 hidden units and hyperbolic tangent nonlinearities. On a given run, each optimization algorithm was started from the same initial random weights, and run until either the NRMSE dropped below .001 or 500 epochs were reached, whichever came first. The statistics given are based on 100 runs from different initial random weights. For this problem, CubicpropII was the clear winner over Gradient Descent and Quickprop in all categories but one; its worst convergent run took 130 epochs, as compared with Quickprop's **93,** although its best run took only 17 epochs, as compared with

Quickprop's 40.

Table 2 gives corresponding results for the parity-3 problem, using a network with 3 hidden units. The statistics given are based on 50 runs from different initial random weights. Here CubicpropII was the clear winner in all categories.

Table 3 gives corresponding results for the parity-4 problem, using a network with 5 hidden units. In this case, each algorithm was allowed to run for up to 800 epochs. The statistics given are based on 50 runs from different initial random weights. Here Quickprop performed better overall, but worse than CubicpropII in the best and worst run categories.

5 Conclusion

We have presented two optimization algorithms based on cubic curve fitting, and demonstrated the effectiveness of one of them, CubicpropII, in training neural networks. CubicpropII compares favourably with Quickprop in terms of convergence rate and stability, and in addition does not require a growth factor to bound the rate of increase of the weights.

Although CubicpropI and II have been presented in the context of neural network training, they are in fact general optimization methods which may

find application in other areas.

6 Appendix

The cubic fitted by CubicpropI in function space is given by

$$ax^3 + bx^2 + cx + d \tag{18}$$

where

$$a = \frac{(f'_k + f'_{k-1})(w_k - w_{k-1}) - 2(f_k - f_{k-1})}{(w_k - w_{k-1})^3} \tag{19}$$

$$b = \frac{3(f_k - f_{k-1})(w_k + w_{k-1}) + f'_{k-1}(w_{k-1}^2 + w_k w_{k-1} - 2w_k^2) + f'_k(2w_{k-1}^2 - w_k w_{k-1} - w_k^2)}{(w_k - w_{k-1})^3} \tag{20}$$

$$c = \frac{-6(f_k - f_{k-1})w_k w_{k-1} - f'_{k-1} w_k(2w_{k-1}^2 - w_k w_{k-1} - w_k^2) - f'_k w_{k-1}(w_{k-1}^2 + w_k w_{k-1} - 2w_k^2)}{(w_k - w_{k-1})^3} \tag{21}$$

$$d = \frac{f_k w_{k-1}^2(3w_k - w_{k-1}) + f_{k-1} w_k^2(w_k - 3w_{k-1}) - f'_{k-1} w_k^2 w_{k-1}(w_k - w_{k-1}) - f'_k w_{k-1}^2 w_k(w_k - w_{k-1})}{(w_k - w_{k-1})^3} \tag{22}$$

The quadratic fitted by CubicpropII in derivative space is given by

$$ax^2 + bx + c \tag{23}$$

where

$$a = \frac{f'_k(w_{k-1} - w_{k-2}) - f'_{k-1}(w_k - w_{k-2}) + f'_{k-2}(w_k - w_{k-1})}{(w_k - w_{k-1})(w_k - w_{k-2})(w_{k-1} - w_{k-2})} \tag{24}$$

$$b = \frac{(f'_{k-1} - f'_{k-2})w_k^2 + (f'_k - f'_{k-1})w_{k-2}^2 - (f'_k - f'_{k-2})w_{k-1}^2}{(w_k - w_{k-1})(w_k - w_{k-2})(w_{k-1} - w_{k-2})} \tag{25}$$

$$c = \frac{f'_k w_{k-1} w_{k-2}(w_{k-1} - w_{k-2}) - f'_{k-1} w_k w_{k-2}(w_k - w_{k-2}) + f'_{k-2} w_k w_{k-1}(w_k - w_{k-1})}{(w_k - w_{k-1})(w_k - w_{k-2})(w_{k-1} - w_{k-2})} \tag{26}$$

The first derivative of this quadratic evaluated at the current point is given

$$2aw_k + b = \frac{f'_k(w_{k-2}^2 + 2w_k w_{k-1} - 2w_k w_{k-2}) - w_{k-1}^2) - f'_{k-1}(w_k - w_{k-2})^2 + f'_{k-2}(w_k - w_{k-1})^2}{(w_k - w_{k-1})(w_k - w_{k-2})(w_{k-1} - w_{k-2})} \tag{27}$$

References

[1] Scott E. Fahlman, "Faster learning variations on back-propagation: An empirical study", in Proceedings of *the 1988* Connectionist *Models* ,Summer School, pp. 38–51. Morgan Kaofmann, June 1988.

[2] Scott E. Fahlman, "An empirical study of learning speed in back-propagation networks", Technical Report CMU-CS-88-162, Carnegie-

Mellon University, June 1988.

[3] D.E. Luenberger, Linear and *Nonlinear Programming*, Addison-VVesley, 1984.

[4] Antonio G. Thome, Massively *Parallel Nonlinear* System Identification *Techniques:* A Committee Architecture, PhD thesis, Purdue University, 1993.

Tahle 1: Statistics computed over 100 runs for Parity-2 using a network with 2 hidden units and hyperbolic tangent nonlinearities.

	Gradient Descent	Quickprop	Cubicprop II
Average over all runs (epochs)	500	163.67	124.85
Std dev over all runs (epochs)	0	184.95	160.84
Median over all runs (epochs)	500	67	65
Average over convergent runs (epochs)	500	63.20	58.64
best run (epochs)	500	[illegible]	17
worst convergent run (epochs)	500	[illegible]	130
Number of non-convergent runs	100	23	[illegible]
Percentage of non-convergent runs	100	23	[illegible]
Average NRMSE	.1260	.1425	.0937

Tahle 2: Statistics computed over 50 runs for Parity-3 using a network with 3 hidden units and hyperbolic tangent nonlinearities.

	Gradient Descent	Quickprop	Cubicprop II
Average over all runs (epochs)	500	63.34	50.06
Std dev over all runs (epochs)	0	21.72	23.31
Median over all runs (epochs)	500	60	44
Average over convergent runs (epochs)	500	G3.34	50.06
best run (epochs)	500	43	18
worst. convergent run (epochs)	500	197	98
Number of non-convergent runs	50	0	0
Percentage of non-convergent runs	100	0	0
Average NRMSE	.0278	.0009	.0009

Table **3:** Statistics computed over 50 runs for Parity-4 using a network with 5 hidden units and hyperbolic tangent nonlinearities.

	Gradient Descent	Quickprop	Cubicprop II
Average over all runs (epochs)	800	327.46	443
Std dev over all runs (epochs)	0	261.22	287.91
Median over all runs (epochs)	800	193	272
Average over convergent runs (epochs)	800	209.32	224.23
best run (epochs)	800	128	86
worst convergent run (epochs)	800	660	389
Number of non-convergent runs	50	10	19
Percentage of non-convergent runs	100	20	38
Average NRMSE	.1275	.0703	.1609

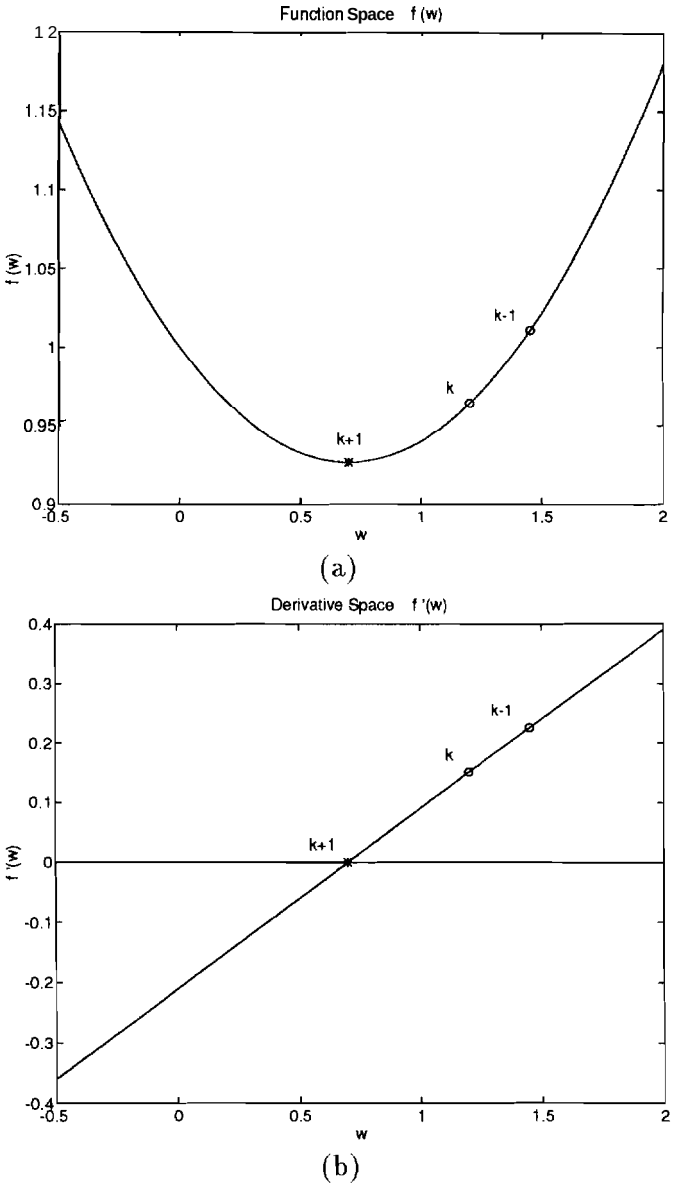


Figure 1: The Method of False Position (aka Quickprop) can be viewed as (a) minimizing a quadratic fitted in function space, or (b) finding the zero of a line fitted in derivative space.

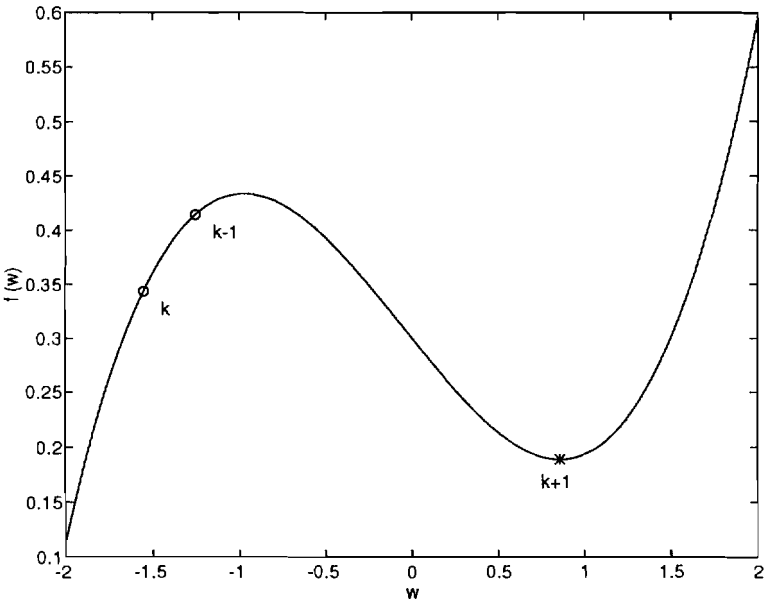
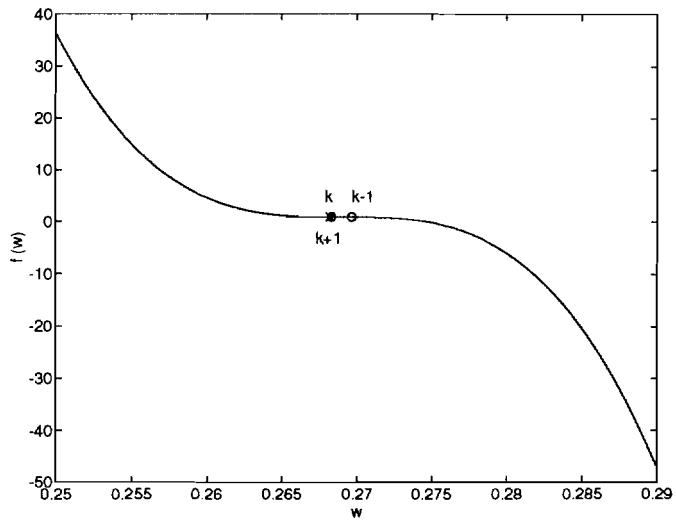
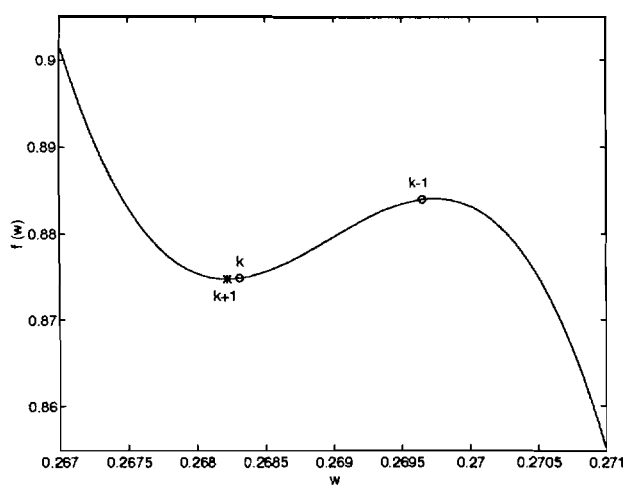


Figure 2: An example in which the minimum of the cubic fitted by CubicpropI (5) is not in the negative gradient direction relative to the current point k.

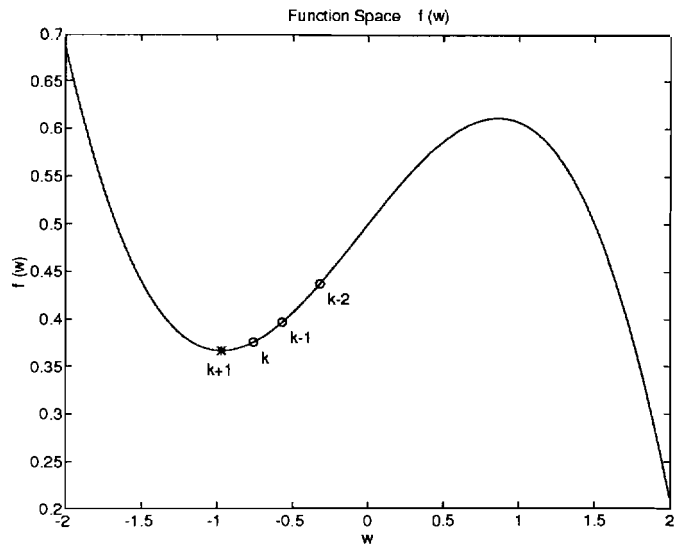


(a)

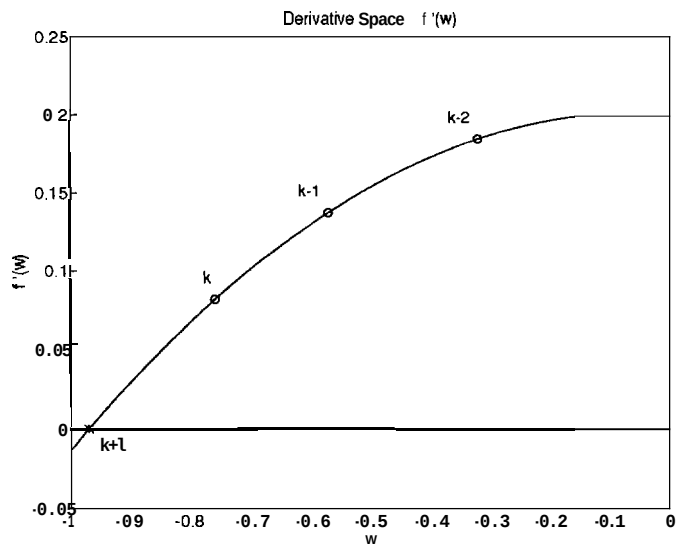


(b)

Figure **3:** (a) A typical cubic fitted by CubicpropI *(5)*. (b) The same curve at a higher magnification.



(a)



(b)

Figure 4: CubicpropII (8) can be viewed as (a) finding the local minimum of a cubic fitted in function space, or (b) finding the zero of a quadratic fitted in derivative space.