MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1433 June, 1993

Measure Fields for Function Approximation

Jose L. Marroquin

Abstract

The computation of a piecewise smooth function that approximates a finite set of data points may be decomposed into two decoupled tasks: first, the computation of the locally smooth models, and hence, the segmentation of the data into classes that consist on the sets of points best approximated by each model, and second, the computation of the normalized discriminant functions for each induced class (which may be interpreted as relative probabilities). The approximating function may then be computed as the optimal estimator with respect to this measure field.

For the first step, we propose a scheme that involves both robust regression and spatial localization using Gaussian windows. The discriminant functions are obtained by fitting Gaussian mixture models for the data distribution in the boundary of each class. We give an efficient procedure for effecting both computations, and for the determination of the optimal number of components. Examples of the application of this scheme to image filtering, surface reconstruction and time series prediction are presented as well.

This report describes research done at the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041; and by a grant from the National Institutes of Health under contract NIH 2-S07-RR07047. Additional support is provided by the North Atlantic Treaty Organization, ATR Audio and Visual Perception Research Laboratories, Mitsubishi Electric Corporation, Sumitomo Metal Industries, and Siemens AG. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. J.L. Marroquin was supported in part by a grant from the Consejo Nacional de Ciencia y Tecnologia, Mexico.

1 Introduction

This paper addresses the problem of approximating a function $z : \mathbf{R}^d \mapsto Q$, given a finite set of data points (examples) $\{(x_i, z_i), x_i \in \mathbf{R}^d, z_i \in Q, i = 1, \ldots, N\}$. The set Q may be a finite set, as in pattern classification problems, or may coincide with the real numbers (as in function approximation, regression and filtering).

A natural way of approaching this problem is to fit simple local models to the data, which are then put together using appropriate weight functions. The fitting process, for each local model, involves: first, the assignment of a relative weight to each data point that indicates its importance for the determination of the model, and second, the computation of the parameters that minimize an appropriate weighted error measure.

These two steps may be carried out in a decoupled fashion, by making the weights depend only on the spatial distribution of the independent variables $\{x_i\}$. Extreme instances of this strategy are the local learning algorithms proposed by Vapnik [1], where a different local model is used for each point x in which the function is to be evaluated, and the model parameters are made to depend only on those data points that are closest to x. Another example may be found in the use of Radial Basis Functions with fixed knots for function approximation; in this case, the location of the knots (and hence, the relative data weights) are determined as a function of the distribution of $\{x_i\}$, for example, using the K–Means algorithm [15].

This decoupled scheme, although computationally simple, does not take into consideration the structure of the function itself, and therefore, will lead to relatively complex local models, so that the approximation will be, in general, difficult to interpret; if the local models are to reflect the structure of the function, the data weights must depend also on the goodness of fit.

A simple —although computationally expensive— way of doing this is to specify parametric models for both the weights and the local regressions, and then adjust the parameters in a coupled way by minimizing an appropriate cost function. Thus, for example, in [16], the cost function is the log–likelihood of a statistical mixture model, which is minimized using gradient descent; the local models ("experts") are multi–layer perceptrons, and the weights ("gating networks") are linear functions followed by an exponential transformation and normalization ("softmax units"). The number of components is found by setting it to a sufficiently large number, and deleting, upon convergence, those components with insignificant weights.

The interpretability of the approximation increases if the local models are given a simpler form. Also, it is computationally more efficient to start with a single component, and successively add new ones by splitting the domain of the one with worst local fit until the global fit error is below a given threshold. This is done, for example, in Recursive Partition Regression (RPR) [2], where the local models are low–order polynomials and the weights are step functions on each one of the independent variables.

The problems with this approach are that the approximating functions are always discontinuous, and the discontinuities are always piecewise parallel to the coordinate axes, so that continuous functions, or piecewise smooth functions with complex discontinuity shapes are not well approximated. This is remedied in the MARS approach [5], by replacing the step functions by truncated power spline basis functions, and by not removing the parent basis functions after they are split. The problem here is that now the resulting approximation is aways globally smooth (so that discontinuous functions are not well approximated), and that the representation can no longer be expressed as a sum of spatially localized simple models, which has a high intuitive appeal.

The objective of this work is to overcome some of these limitations; specifically, our goal is to develop a representation that is : easy to interpret; efficient to compute, and flexible, in the sense that it can generate both smooth and piecewise smooth approximations. It is based in the following ideas:

The most flexible representation has the form of a *measure field*, which is defined by the following elements:

i) A relatively small set of simple functions, $\{\hat{z}_k : \mathbf{R}^d \mapsto Q, k = 1, \ldots, m\}$ (the local models).

ii) A probabilistic description of the regions where each local model is a good approximation of the function, that is, a set of functions $\{\rho_k : \mathbf{R}^d \mapsto [0, 1], k = 1, \ldots, m\}$ such that $\sum_k \rho_k(x) = 1.0$ for all x, where $\rho_k(x)$ represents the probability that the function is optimally approximated by $\hat{z}_k$ at location x.

The pairs $\{(\rho_k, \hat{z}_k)\}$ define a discrete *measure field* p that represents the function:

$$p_x(z) = \sum_{k=1}^{m} \rho_k(x) \delta(z - \hat{z}_k(x)) \qquad (1)$$

where $\delta(\cdot)$ is the usual delta function.

One may specify different criteria for computing the approximation $\hat{z}$ given the measure field (1). For example, a smooth function may be obtained by taking the mean value:

$$\hat{z}(x) = \int z dp_x(z) = \sum_{k=1}^{m} \rho_k(x) \hat{z}_k(x) \qquad (2)$$

and a piecewise smooth one by taking the mode:

$$\hat{z}(x) = \hat{z}_r(x) \ , \ r = \arg\max_k \rho_k(x) \qquad (3)$$

It is also possible to include additional information, such as prior conditions on the shape of the smooth patches or on the location of the discontinuities, and compute $\hat{z}$ as the optimal Bayesian estimator [12].

This type of representation has been proposed in the context of machine learning [16] [8], and also for the solution of a class of computational vision problems [12]. It will be adequate if the models for the functions $\hat{z}$ and ρ are intuitively appealing, and allow for a computationally efficient implementation. Here, we will use for the local models polynomials of low order (0 or 1), so that piecewise constant and piecewise linear functions are exactly represented. The discriminant functions may be modeled as normalized sums of Gaussians:

$$\rho_k(x) = \frac{\sum_{j=1}^{n_k} G_{kj}(x)^\beta}{\sum_{r=1}^{m} \sum_{s=1}^{n_r} G_{rs}(x)^\beta} \tag{4}$$

where: $\{G_{kj}(\cdot)\}$ are Gaussians; β is a positive number that controls the smoothness of the approximation, and n_k is the number of Gaussians that model the distribution of class k.

For the computation of the corresponding parameters, we propose a strategy that combines the computational efficiency of decoupling the weight and model computation with the power of making the weights dependent on the local fit. The scheme is as follows: compute the local models in a first step using weights that depend on both x and z, that is, using local *robust* regression; in a second step, classify the data according to the model that best approximates the function at each point, and finally compute the ρ functions as the discriminants for this induced classification task (for classification problems, of course, the first 2 steps are not needed).

The number of components is determined by a strategy similar to that of RPR; first, components are added one at a time, until the error variance reaches an appropriate value. In a subsequent "backwards" step, redundant components are merged (in the regression phase) or deleted (in the classification phase).

What makes this methodology attractive is the existence of efficient algorithms to perform the computations. In the next section we will show that by using Gaussian windows, local robust regression becomes equivalent to the estimation of the parameters of a Gaussian mixture model (like the one used in the classification stage), and that this problem is equivalent to vector quantization with a quadratic distortion measure, for which efficient algorithms exist. Since the discriminant functions in the classification phase are also modeled as Gaussian mixtures, a single computational framework may in fact be used for both steps.

2 Gaussian Mixture Models and the Generalized K-Means Algorithm

Suppose we have N data points in $\mathbf{R}^d$, $\{x_i, i = 1, \ldots, N\}$, and we want to approximate their distribution with n codewords or "centers" $\{m_k \in \mathbf{R}^d, k = 1, \ldots, n\}$ that minimize a distortion measure of the form:

$$\mathcal{E} = \sum_{i=1}^{N} \sum_{k=1}^{n} d(x_i, m_k) s_{ik} \tag{5}$$

where $d(x_i, m_k)$ is a local distortion measure, such as:

$$d(i,k) = ||x_i - m_k||^2 \tag{6}$$

The indicator variables s_{ik} are given by:

$$s_{ik} = \mathbf{1}_{S_k}(x_i)$$

with $\mathbf{1}_S$ denoting the indicator function of set S, and where the sets S_k consist on those data points that are mapped into codeword k.

It is possible to show [11] that the following (K–Means) algorithm will never increase $\mathcal{E}$:

```
1: for $k = 1, \ldots, n$, assign to each $m_k^{(0)}$ a random location ;
2: at time $t$, set
     $s_{ik}^{(t)} = 1$ , if $d(x_i, m_k^{(t)}) < d(i, m_j^{(t)})$ , $j \neq k$
             $= 0$ , otherwise
   (with an appropriate rule for solving the ties) and
     $m_k^{(t+1)} = \arg\min_u \sum_{i=1}^{N} \sum_{k=1}^{n} d(x_i, u) s_{ik}^{(t)}$     (7)
   for example, if $d(\cdot,\cdot)$ is given by (6), we get:
     $m_k^{(t+1)} = \frac{\sum_i x_i s_{ik}^{(t)}}{\sum_i s_{ik}^{(t)}}$
```

Alternatively, one may consider the data as samples from a mixture of populations $\{S_k, k = 1, \ldots, n\}$, each one normally distributed with mean m_k and identical covariances $\frac{1}{\beta}I$.

Assuming that for each i the indicator variables for these populations, $\{s_{ik}\}$, are independently and identically distributed according with a multinomial distribution consisting on one equiprobable draw on n categories, the conditional log density for data point x_i is thus:

$$\log p(x_i \mid s) = -\beta \sum_{k=1}^{n} s_{ik} ||x_i - m_k||^2 + K$$

where K is a constant.

Assuming that the data $x = \{x_i\}$ given $s = \{s_{ik}\}$ are conditionally independent, we get that the log likelihood for the complete data x and s is:

$$\mathcal{L}(m) = -\beta \sum_{k=1}^{n} \sum_{i=1}^{N} s_{ik} ||x_i - m_k||^2 + NK \tag{8}$$

We may now find the maximum likelihood estimator for $m = \{m_k\}$ by treating s as missing data and applying the EM algorithm [4] [7]: in the expectation (E) step at time t, the expected likelihood $< \mathcal{L} >$ is found by replacing the variables $\{s_{ik}\}$ in (8) with their conditional expected value, given x and $m^{(t)}$. This is found using Bayes rule:

$$E[s_{ik} \mid x, m^{(t)}] \doteq w_{ik}^{(t)} = \Pr(x_i \in S_k \mid x_i, m^{(t)}) = \frac{\exp[-\beta ||x_i - m_k^{(t)}||^2]}{\sum_{r=1}^{n} \exp[-\beta ||x_i - m_r^{(t)}||^2]}$$

In the M step, we find the value of m that maximizes $< \mathcal{L} >$:

$$m_k^{(t+1)} = \arg\min_{m_k} \sum_i w_{ik}^{(t)} ||x_i - m_k||^2 = \frac{\sum_i w_{ik}^{(t)} x_i}{\sum_i w_{ik}^{(t)}}$$

In the limit as $\beta \to \infty$, each indicator variable is replaced by the mode of the conditional distribution (which coincides with the mean) in the E step, and so, the EM algorithm reduces to the K–Means scheme. For finite values of β, it minimizes a distortion measure that is obtained from the log likelihood of the data:

$$\mathcal{E} = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{n} \exp[-\beta \|x_i - m_k\|^2] \right)$$

In practice, the behavior of both algorithms is very similar, although it has been reported that the EM scheme tends to produce more uniform center distributions [16]. If one uses the EM algorithm, one can estimate, upon convergence, the indicator variables s_{ik} by:

$$\begin{aligned} s_{ik} &= 1 \text{ , if } w_{ik} \geq w_{ij} \text{ for all j} \\ &= 0 \text{ , otherwise} \end{aligned} \tag{9}$$

and thus define the domain for each center k as the set of points x_i for which $s_{ik} = 1$.

Suppose now that in addition to the data points $\{x_i\}$ we are given the corresponding values $\{z_i\}$ of a function that we want to approximate. The problem now is to find not only the set of centers $\{m_k\}$ that represents the data, but also to associate with each one of them a local model $\hat{z}_k(x; \theta_k)$ that represents the function z with minimum distortion in the domain (Voronoi polytope) of the corresponding center. This means that we now have to minimize a distortion measure of the form:

$$\mathcal{E} = \sum_{i=1}^{N} \log \sum_{k=1}^{n} \exp - \left[\beta \left[\|x_i - m_k\|^2 + \lambda \Phi(x_i, z_i; \theta_k)\right]\right] \tag{10}$$

where: Φ measures the fit error of the model given by the parameter vector θ_k (and possibly additional prior contraints on this vector), and λ is a parameter.

To derive the generalized K-Means (GKM) algorithm in this case, we consider a mixture model for the joint distribution of x and z and obtain the GKM scheme as the limit, as $\beta \to \infty$ of the EM algorithm that estimates the vector (m, θ); the main iteration takes now the form:

1. E Step: compute

   $$w_{ik}^{(t)} = \frac{\mathcal{G}_k(x_i)}{\sum_r \mathcal{G}_r(x_i)}$$

   where

   $$\mathcal{G}_k(x_i) = \exp\left[-\beta[\|x_i - m_k^{(t)}\|^2 + \lambda \Phi(x_i, z_i; \theta_k^{(t)})]\right]$$

   which in the GKM case reduces to:

   $$\begin{aligned} w_{ik}^{(t)} &= 1 \text{ , if } \mathcal{G}_k(x_i) > \mathcal{G}_r(x_i), r \neq k \\ &= 0 \text{ , otherwise} \end{aligned}$$

2. M Step: Set

   $$\begin{aligned} &(m_k^{(t+1)}, \theta_k^{(t+1)}) = \\ &= \arg\min_{u,v} \sum_i w_{ik}^{(t)} \left(\|x_i - u\|^2 + \lambda \Phi(x_i, z_i; v)\right) \end{aligned}$$

The choice of the function Φ is determined by the desired behavior of the centers, and also by computational considerations: if it is quadratic in θ, the maximization steps in the above algorithms are implemented simply by matrix inversion. We now discuss the specific choices that are relevant for the determination of the local models and for the classification stage.

2.1 Local Robust Regression

In this case, a natural choice for Φ is the quadratic fit error; if the local models are linear, so that $\hat{z}_k = a_k \cdot x + b_k$, we get:

$$\Phi(x, z; \theta_k) = (z - a_k \cdot x - b_k)^2 \tag{11}$$

The function Φ may be modified to enforce some prior constraint; for example, to give higher preference to horizontal planes (i.e., for local ridge regression [22]) we may put:

$$\Phi(x, z; \theta_k) = (z - a_k \cdot x - b_k)^2 + \nu \|a_k\|^2$$

With this choice, the M step of the GKM (or EM) algorithms is equivalent to the solution of a decoupled set of linear systems (one for each local model) whose size grows linearly with the dimension of the input space.

The solutions found by these algorithms may be understood, in terms of robust regression, as the best n hyperplanes that pass through the data, with the constraint (enforced by the $\|x - m_k\|^2$ terms) that the support for each hyperplane must be spatially localized; this constraint is crucial to get good results, since it prevents the system from finding suboptimal solutions with non–local support. The relative weight of this constraint is controlled by the parameter λ; if its value is too high, the non–local solutions will prevail, while if it is too low, the location of the centers will be controlled solely by the data density. The solution, however, is not very sensitive to the precise value of this parameter; if one scales the data so that all x_i are inside the unit hypercube, and all z_i are in the interval $[0, 1]$, a value of λ between 1 and 5 gives good results.

The nature of the solution also depends on the number of local models: there should be enough of them for the system to escape local minima that correspond to non–locally supported hyperplanes. On the other hand, if there are too many, one would be, in effect, modeling the noise (i.e., overfitting the data) with the corresponding degradation of the generalization capabilities of the system.

This trade–off between model capacity and generalization error has been widely recognized, and criteria different from the minimization of the empirical risk have been proposed to find an optimal compromise (e.g., minimization of the structural risk [1]; minimum description length criteria [20] or cross validation techniques [3]).

In this case, due to the simplicity of the local models, one may use directly as a criterion an approximation to the expected value of the fit error. In [6] it is shown that this expected value may be decomposed into a bias and a variance terms:

$$E[(\hat{z}(x) - z(x))^2] = (E\hat{z}(x) - z(x))^2 + E[(\hat{z}(x) - E\hat{z}(x))^2]$$

If $z(\cdot)$ is piecewise linear, and there are enough local models, the corresponding piecewise linear regression $\hat{z}(x)$ is unbiased, so that the mean fit error is controled by the variance. For each local model, the variance of the predicted mean response $\hat{z}_k(p)$, for any point p inside its Voronoi polytope S_k, may be estimated as [21]:

$$V_k(p) = \text{Var}(\hat{z}_k(p)) = \sigma_k^2\left(\frac{1}{n_k} + (p - \bar{x}_k)^T (X_k^T X_k)^{-1} (p - \bar{x}_k)\right) \tag{12}$$

where n_k is the number of examples inside S_k;

$$\bar{x}_k = \frac{1}{n_k} \sum_{i:x_i \in S_k} x_i;$$

$$\sigma_k^2 = \frac{1}{n_k - D - 1} \sum_{i:x_i \in S_k} (z_i - \hat{z}_k(x_i))^2$$

where D is the dimension of the input space and the $(n_k \times D)$ matrix X_k whose rows are $(x_i - \bar{x}_k)^T$ for each $x_i \in S_k$.

Equation (12) allows one to estimate the reliability of the approximation at any point; therefore, one may use it to estimate, in addition to the reconstructed surface, a corresponding "error map".

The optimal number of models is determined by minimizing the average variance. This may be approximated by the mean variance at the centers $\{\bar{x}_k\}$ of the Voronoi polytopes: averaging $V_k(\bar{x}_k)$ over all K centers, and noting that $n_k \approx N/K$, we get:

$$\bar{V}_K \approx K\sigma^2/N$$

where

$$\sigma^2 = \frac{1}{N} \sum_{k=1}^{K} \sum_{i:x_i \in S_k} (z_i - \hat{z}_k(x_i))^2$$

is the empirical risk.

The complete strategy, therefore, consists in adding one model at a time (e.g., by splitting the center with worst local fit error) until a maximum number K_{max} is reached, and then pick the solution that corresponds to the number of centers K that minimizes $\bar{V}_K$. The number K_{max} is related to the minimum number of examples that reliably support a hyperplane, and may depend on: the dimension of the input space; the total number of examples and the noise power. Although it is convenient to choose it in a reasonable way, its precise value is not critical for the performance of the system.

Since the models are locally supported, it is possible that several of them correspond in fact to the same —or very similar— hyperplanes. For this reason, it is necessary, once a solution is found, to merge together the redundant components. To do this, one should find an appropriate way of measuring how different two hyperplanes are. It may be shown [19] that given the usual normality assumptions, the relative MSE reduction obtained by replacing the two hyperplanes by a single one (with the appropriate corrections to compensate for the different number of parameters) has a standard F distribution and hence, it may be used for testing the corresponding hypothesis. In particular, for each pair of models (j,k) one computes the statistic:

$$\hat{F}_{jk} = \frac{(SS_{jk} - SS_j - SS_k)/(D+1)}{(SS_j + SS_k)/(n_j + n_k - 2(D+1))}$$

where D is the dimensionality of the input space;

$$SS_j = \sum_{i:x_i \in S_j} (z_i - \hat{z}_j(x_i))^2$$

$$SS_{jk} = \sum_{i:x_i \in S_j \cup S_k} (z_i - \hat{z}_{jk}(x_i))^2$$

where $\hat{z}_{jk}$ is the best (LSE) hyperplane through the points in $S_j \cup S_k$.

Each merging step consists in finding the models (j,k) with the smallest value of $\hat{F}_{jk}$; this statistic is then compared with the tabulated value of F with $(D+1)$ and $(n_j + n_k - 2(D+1))$ degrees of freedom at the appropriate confidence level; if it is smaller, the 2 models are replaced by $\hat{z}_{jk}$ and the whole process is repeated until the null hypothesis (equality of the regression hyperplanes) is rejected for the smallest F.

This test may fail if the normality assumption is grossly violated (although we have found that it is relatively robust). In these cases it may be convenient to also put an upper bound on the number of linear components that one wants to include in the complete model, so that the merging process stops only if the smallest $\hat{F}$ is too large and the total number of components is below this bound.

After the merging process is completed, one associates with each data point x_i a class c_i that corresponds to the model that contains the center k for which the indicator variable $s_{ik} = 1$. To compute the desired measure field representation, one now has to find the discriminant functions for the data $\{(x_i, c_i), i = 1 \ldots N\}$.

2.2 Local Gaussian Classifiers

Consider the problem of finding discriminant functions that optimally classify the data: $\{(x_i, c_i), i = 1 \ldots N\}$, $c_i \in \{1, \ldots, Q\}$ for all i. A simple way of doing this is to approximate the data distribution within each class c with a sum of n_c Gaussians whose centers are found again with the GKM algorithm [13]. This may be obtained from the above model by setting λ to a very large value, and Φ to:

$$\Phi(x, z; m_{kc}) = 1 - \delta(z(x) - c)$$

where m_{kc} is the center of the k^{th} Gaussian that approximates the data distribution of class c.

One may now define a 1–parameter family of discriminant functions (indexed by the positive parameter β) as:

$$\rho_c(x) = \frac{\sum_{k=1}^{n_c} G_{kc}(x)^\beta}{\sum_{q=1}^{Q} \sum_{r=1}^{n_q} G_{rq}(x)^\beta} \qquad (13)$$

where

$$G_{kc}(x) = \mid A_{kc} \mid^{-1/2} \exp[-\frac{1}{2}(x - m_{kc})^T A_{kc}^{-1}(x - m_{kc})]$$

The matrices A_{kc} ($\mid A_{kc} \mid$ denotes their determinants) may all be taken equal to the identity (in which case this procedure becomes similar to the LVQ1 method [9]), or they may be computed as the covariance matrices inside the corresponding Voronoi polytope (see [13]):

$$A_{kc} = \frac{1}{n_{kc}} \sum_{x_i \in S_{kc}} (x_i - m_{kc})(x_i - m_{kc})^T$$

(this is usually important in high dimensions with anisotropic data).

For high values of β, the classifier simply assigns to each point the class of the closest center (with a distance weighted by A matrices). As β decreases, the discriminant functions —and hence the inter–class boundaries— become smoother, so that β controls, in some sense, the trade–off between capacity and generalization capabilities.

If the classifier is trained in such a way that it achieves (near) perfect classification of the data with high β, this parameter may then be used to fine–tune the classifier (using, for example cross–validation) to attain low generalization errors.

Training the classifier means finding the minimal number of centers that achieve a given classification error. For this, one may use the following procedure: start with one center per class; upon convergence of the K–Means procedure, compute the approximate classification; add a new center at a location that corresponds to a misclassified data point in the class with highest classification error, and repeat the K–Means process.

Once the target approximation accuracy (or the maximum number of allowable centers) is reached, one should "prune" the irrelevant centers (those sample the interior of a class, and therefore do not contribute to the correct classification of any point). More precisely, a center m_{kc} is pruned if its irrelevance function $I(k,c)$ equals 1. $I(k,c)$ is defined as:

$$I(k,c) = \delta\left(\sum_{i:\hat{z}(x_i)=c} (1 - \delta(c - \tilde{z}_{kc}(x_i))\right)$$

where $\hat{z}(x_i)$ is the current estimate for the class of x_i, and $\tilde{z}_{kc}(x_i)$ is the class that would be estimated if m_{kc} was deleted. Note that for high β one needs to consider in the sum only those points in the Voronoi polytope of m_{kc}, and $\tilde{z}_{kc}(x_i)$ is simply the class associated with the second nearest center to x_i.

In practice, it is possible to prune the irrelevant centers dynamically by computing $I(k,c)$ for all centers at each iteration; if a center is found to be irrelevant, it is pruned and all the data points inside its Voronoi polytope are marked as inactive. In this way, one works with less and less data points, accelerating the whole process (provided that at each iteration one checks the inactive points and includes again those that are misclassified). We have found that in general this procedure gives the same results as the one described above, except when there is a very large inter–class overlap (classification noise), when the dynamic pruning may fall into a limit cycle.

An interesting by–product of this pruning procedure is a secondary class that may be associated with each data point, namely, an indicator of its relevance or irrelevance for the primary classification task. One may then construct a binary classifier for this secondary class which may be useful for guiding the recollection of new data, particularly when this involves complex and expensive procedures.

3 Examples

In this section we present some examples that illustrate the performance of these techniques.

The first set of examples deal with 2–dimensional problems. The first one shows that the system correctly identifies the piecewise linear structure when it is present: panel (a) of figure 1 shows a function $z(x,y)$ that is equal to a tilted plane inside an L–shaped region and is constant elsewhere. The input to the system consisted on 200 triplets $\{(x_i, y_i, z_i)\}$, where each (x_i, y_i) is a randomly selected point on the square and $z_i = z(x_i, y_i) + n_i$, where $\{n_i\}$ is a set of independent, identically distributed Gaussian random variables with zero mean and variance equal to 0.05. In step 1, the algorithm found 5 local models, which after the merging step were reduced to 2. The second step was thus a binary classification problem for which perfect classification was achieved with 26 Gaussians (with covariance matrices equal to the identity), which after pruning were reduced to 17. The reconstructed surface, obtained as the mode of the measure field with $\beta = 8$, is shown in panel (b). Note that additional information about the inter–class boundary may be easily incorporated, improving the solution without having to recompute the measure field; for example, if one wishes to reconstruct the surface in the nodes of a square lattice L, one may include a smoothness constraint on these boundaries in the form of a prior Markov Random Field (MRF) model for the shape of the smooth patches (see [12]). Using, for instance, a first order MRF model with Ising potentials, one may obtain the optimal classification $\{\hat{c}_i, i \in L\}$ as the MPM estimator [14] with respect to the posterior Gibbs distribution with energy:

$$U(\hat{c}) = \sum_{[i,j]} V(\hat{c}_i, \hat{c}_j) - \sum_{i \in L} \rho_{\hat{c}_i}(xl(i))$$

with

$$\begin{aligned} V(a,b) &= -\gamma \text{ , if } a = b \\ &= \gamma \text{ , otherwise} \end{aligned}$$

where: $[i,j]$ denotes summation over all nearest neighbor pairs of sites in L; γ is a parameter and $xl(i)$ is the coordinate vector of node i.

Figure 1–c shows the top view of the "true" classification; panels (d) and (e) show the maximum likelihood classifications (obtained by selecting the class with largest ρ_c at each node) for $\beta = 8$ and $\beta = 2$, respectively. The optimal Bayesian (MPM) classification (with $\gamma = 10$) is shown in panel (f). Note that in all cases all the examples are correctly classified; the generalization performance, however, will clearly be superior in the last case.

Figure 1 around here

The second example illustrates the system performance when one of the surface patches is non–linear. For this, we use the function studied in [10], which is illustratred in panel (a) of figure 2 on the unit square. Again, the input to the system consisted in 200 triplets $\{(x_i, y_i, z_i)\}$ with (x_i, y_i) selected randomly. The system found 6 components in step 1 (after merging), and 21 Gaussians (with covariance matrices equal to the identity) in the 6–way classifier of step 2. The reconstructed surface is now computed as the mean of the measure field (to get a smooth reconstruction). It is shown in panels (b) and (c) of figure 2 for $\beta = 10$ and $\beta = 2$, respectively.

Figure 2 around here

3.1 Image Filtering and Segmentation

An important class of problems arises when one has dense, noisy data on a regular lattice (i.e., an image), so that no interpolation is needed. In this case, the desired result is just the output of step 1. This output may be interpreted both as a filtered image (since the fitted local models smooth out the noise) and as a segmentation, where each segment corresponds to the domain of each local model. Since there is an observation z_i at every point where one wants to compute the approximation, one may also construct a measure field representation by putting:

$$\rho_k(x_i) = \frac{\exp\left[-\beta[\|x_i - m_k\|^2 + \lambda(z_i - \hat{z}_k(x_i))^2]\right]}{\sum_j \exp\left[-\beta[\|x_i - m_j\|^2 + \lambda(z_i - \hat{z}_j(x_i))^2]\right]}$$

for each center k (note that after merging one may have $\hat{z}_k(x) = \hat{z}_j(x)$ if centers j and k belong to the same component). By setting the filtered image to the mean value with respect to this measure field, one may use β to control the amount of inter-component smoothing: for large β we get an edge-preserving filter and for small β a global smoother. The performance of this scheme for a piecewise linear, noisy synthetic image is shown in figure 3. Note the correct identification of the 3 linear components (after merging), and hence, the correct segmentation, in spite of the fact that the intensity gradient between linear regions disappears in some parts of the image. The shape of the inter-class boundaries may be made smoother (and the isolated pixels eliminated) by using a Gibbsian prior and computing the MPM estimator as above.

Figure 3 around here

A similar idea may be used for image quantization; suppose one wishes to represent an image with a fixed, small number of grey levels. In this case, the local models are 0-order polynomials (constants) and the merging phase is stopped when the desired number of components is reached. The result is illustrated in figure 4, where a detail of a real, noisy image is quantized to 5 levels (a uniform quantization is also included for comparison).

Figure 4 around here

3.2 Time Series Prediction

This example illustrates the system performance in more than 2 dimensions and in a situation where the basic model assumptions (piecewise linear structure of the true function and additive Gaussian noise) are grossly violated; as we show, the performance is still reasonable in this case.

The task is to predict the value of a time series at some future time, based in 4 previous values. To be able to make comparisons with other approximation methods, we use the same example as in [17]: the chaotic time series created by the Mackey–Glass delay–difference equation:

$$y(t+1) = (1-b)y(t) + a\frac{y(t-\tau)}{1+y(t-\tau)^{10}}$$

with $a = 0.2$, $b = 0.1$ and $\tau = 17$.

Each 4-dimensional example point x_i is formed by the values of the series at times: $T_i - 6$, $T_i - 12$, $T_i - 12$ and $T_i - 18$. The corresponding z_i is the value at $T_i + 85$. Using a training set of 1000 examples the local robust regression finds 26 components, which after merging are reduced to 21. For the classification phase, a simple Gaussian classifier (i.e., with one center per class and $\beta = 1$) was used (giving 3.8 % classification error). The final normalized rms error is 0.12, which is comparable to the accuracy of a standard Radial Basis Function network [18] performing the same task (with an rms error of 0. 1), as reported in [17].

The most accurate algorithms reported there: a multi-layer perceptron trained with back propagation and a particular Resource-allocating network, achieved a normalized rms of 0.03). It is worth noting, however, that the computational load of our method is very light, and the total number of parameters, which is: $21 \times (4 + 2 + 10) = 336$, makes this representation more compact than all others reported in [17] (all use more than 500 parameters).

4 Summary and Discussion

We have presented a strategy for approximating a function, given a set of examples, in which an intermediate representation —a measure field— is computed in a first stage. This representation, which consists in a set of simple local models with associated probabilities, has the advantage of allowing for globally or piecewise smooth reconstructions without any additional computational effort; also, it permits the incorporation of additional information about the reconstructed function (e.g., about the location or shape of the discontinuities, etc.) in an easy way.

For the computation of this representation, we proposed a procedure in which the local models are found in a first step, and induce a segmentation of the data (in terms of which local model is supported by each data point). The associated probabilities are then computed in a second step as the discriminant functions of this induced classification task.

This procedure has the advantage of combining the computational simplicity of decoupled methods for finding the domain of each local model, with the power of making the local model selection dependent on the local fit, which permits the use of simpler models that are easy to interpret. What makes it specially attractive is the existence of efficient algorithms for performing these computations; these algorithms are generalizations of the K-Means procedure for vector quantization, with the addition of terms that measure the goodness of fit. Here, they were derived from the EM algorithm for computing the parameters of a Gaussian mixture model.

An important problem that this procedure has in common with all methods that approximate a function as a combination of local models is the trade-off between overall capacity and generalization capabilities. Here, we proposed a simple statistical criterion, namely, the average variance of the mean predicted response of each model, to find the optimal number of models.

It is also important to eliminate redundancies after each step has converged; these redundancies are caused, in the first step, by the fact that the local models are spatially localized (so that a hyperplane that extends over a large region of the input space will be broken into several components to find a good approximation to its boundary). In the second step, there are Gaussians that approximate the data distribution away from the inter-class boundaries and are thus irrelevant for the classification task. In the proposed scheme, these redundancies are eliminated by merging similar hyperplanes (using a criterion based on the F statistic) after the first step, and by pruning irrelevant components after the second; this last method has the advantage of producing a model for the relevance of the examples, which may be used to guide the gathering of additional data.

The results of the experiments we have performed, indicate the plausibility of this approach for general function approximation tasks, and specially for surface reconstruction problems like those that occur in image processing and computational vision.

Acknowledgements: I wish to thank Federico Girosi, Tomaso Poggio and John Harris for many useful discussions. The data for the time series prediction example were kindly provided by John Harris.

References

[1] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, November 1992.

[2] L. Breiman, J. H. Friedman, R.A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[3] P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross validation. *Numer. Math*, 31:377-403, 1979.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. B*, 39:1–38, 1977.

[5] J.H. Friedman. Multivariate adaptive regression splines. *Ann. Stat.*, 19:1-141, 1991.

[6] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[7] McLachlan G.J. and Basford K.E. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, New York and Basel, 1988.

[8] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. F. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[9] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[10] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting.* Academic Press, London, 1986.

[11] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Proceedings of the IEEE*, Com-28(1):84–95, January 1980.

[12] J. L. Marroquin. Random measure fields and the integration of visual information. *IEEE Trans. SMC*, 22:705–716, 1992.

[13] J. L. Marroquin and F. Girosi. Some extensions of the k–means algorithm for image segmentation and pattern classification. AI Memo 1390, MIT Artificial Intelligence Lab., Cambridge, MA, January 1993.

[14] J. L. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *J. Amer. Stat. Assoc.*, 82:76–89, 1987.

[15] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[16] S.J. Nowlan. Soft competitive adaptation. Ph.D. Thesis CMU-CS-91-126, Carnegie Mellon Univ., Pittsburgh, PA, April 1991.

[17] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991.

[18] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

[19] C. R. Rao. *Linear Statistical Inference and its Applications.* Wiley, New York, 1973.

[20] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[21] S. R. Searle. *Linear Models.* J. Wiley, New York, 1971.

[22] H. D. Vinod. A survey of ridge regression and related techniques for improvements over ordinary least squares. *Review of Economics and Statistics*, 60:121–131, February 1978.

FIGURE CAPTIONS

Figure 1: (a) Original surface. (b) Reconstructed surface from 200 examples. (c) True classification (d) Maximum likelihood classification ($\beta = 8$). (e) Maximum likelihood classification ($\beta = 2$). (f) Bayesian (MPM) classification.

Figure 2: (a) Original surface. (b) Reconstructed surface from 200 examples ($\beta = 10$). (c) Same as (b) with $\beta = 2$.

Figure 3: (a) Original noisy image. (b) Filtered image. (c) Segmentation (each grey level corresponds to a linear component).

Figure 4: (a) Original image. (b) Optimal quantization. (c) Uniform quantization.

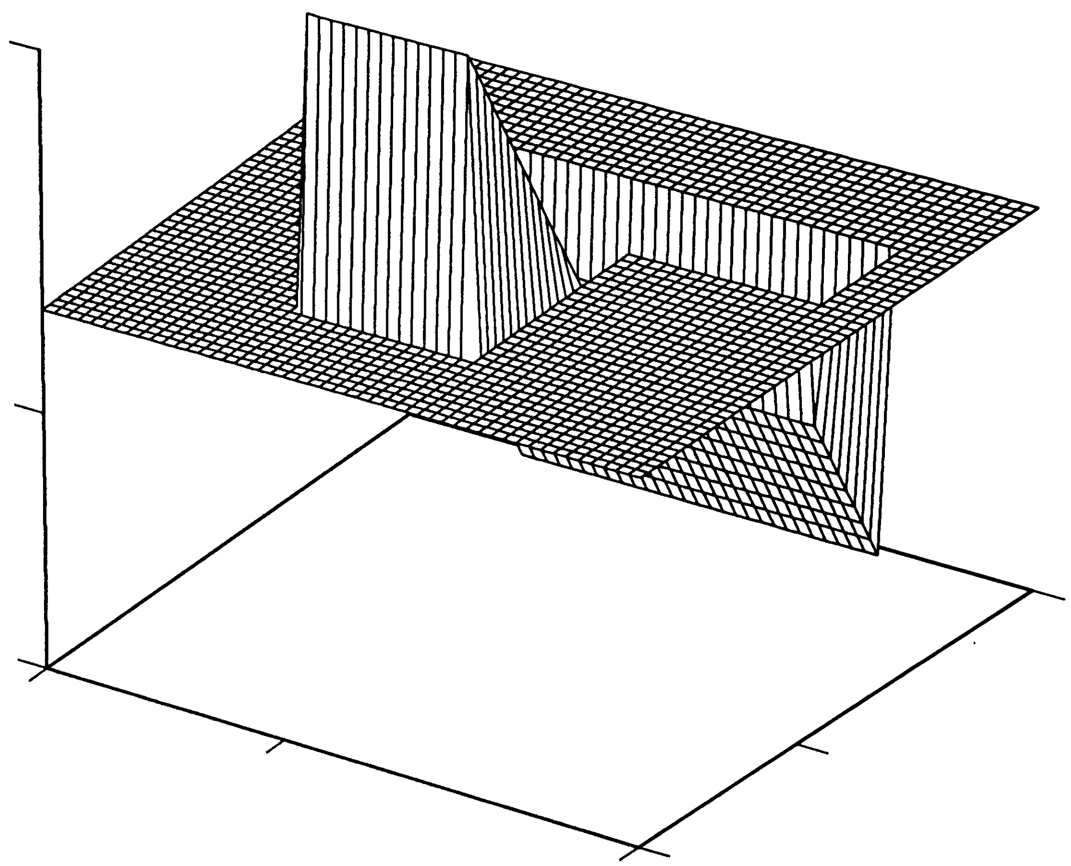


Fig (1-a)

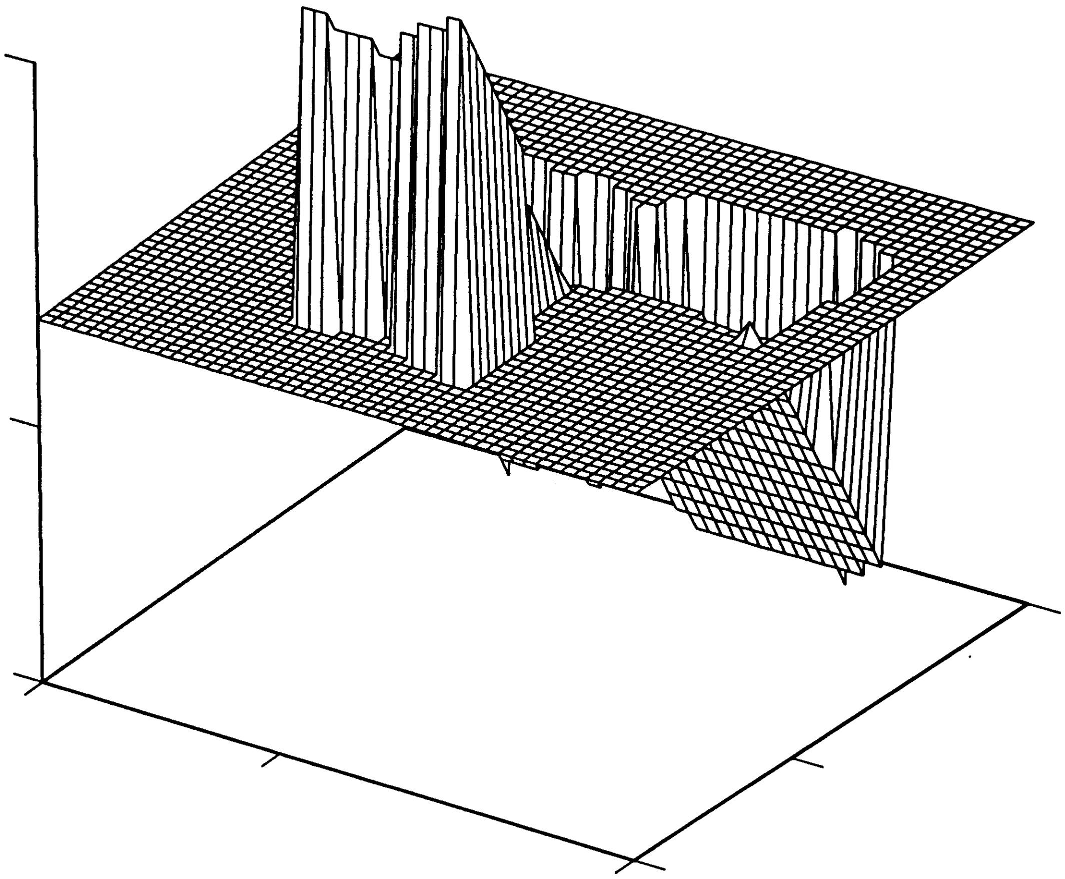
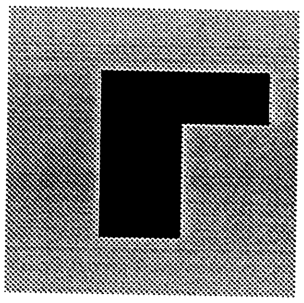
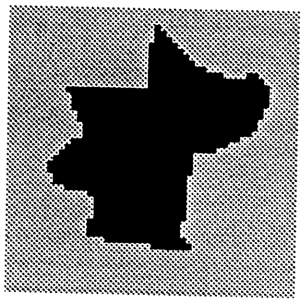


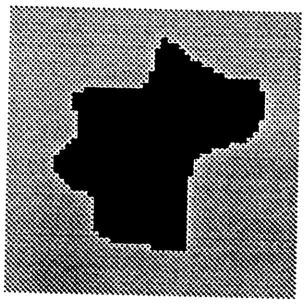
Fig (1-6)



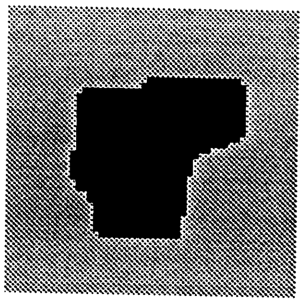
(1-c)



(1-d)



(1-e)



(1-f)

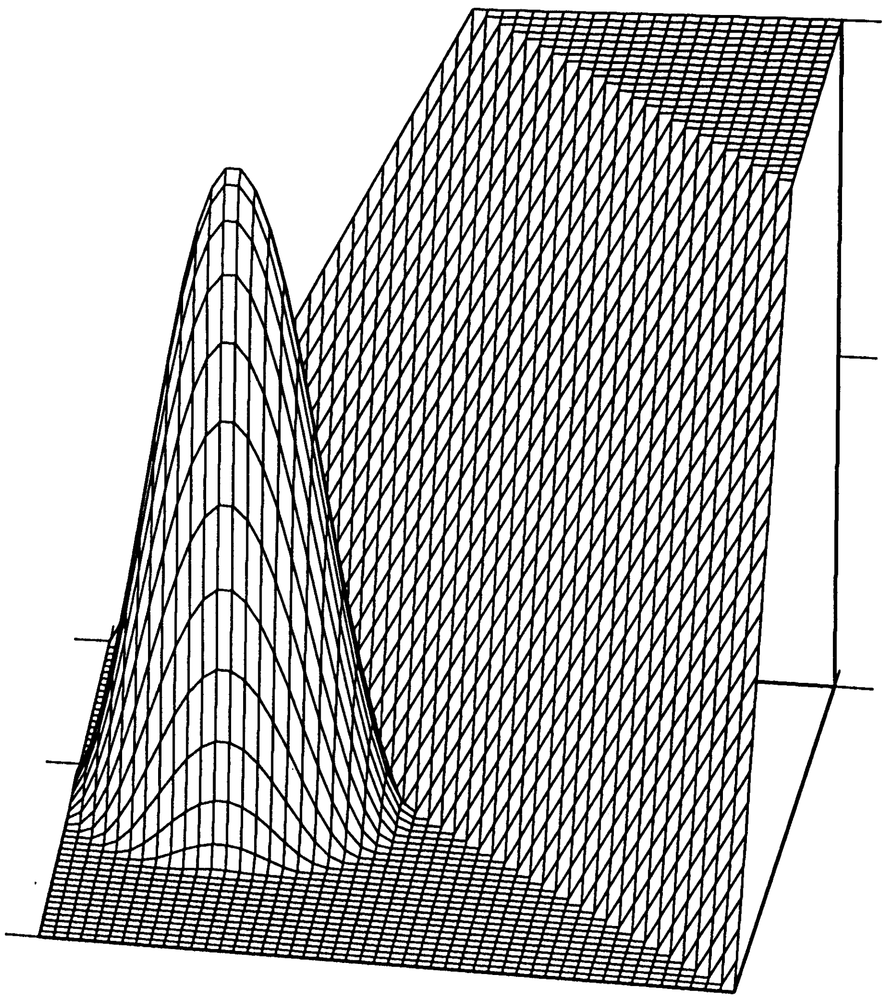
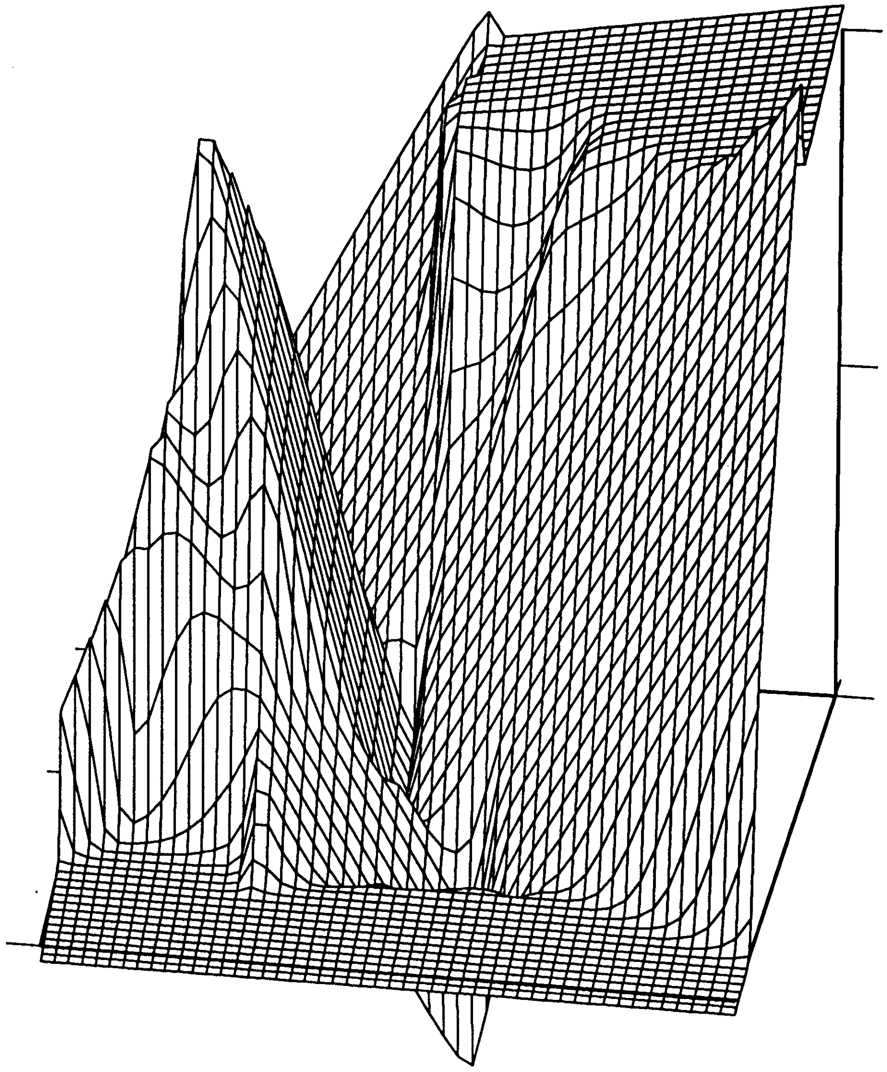
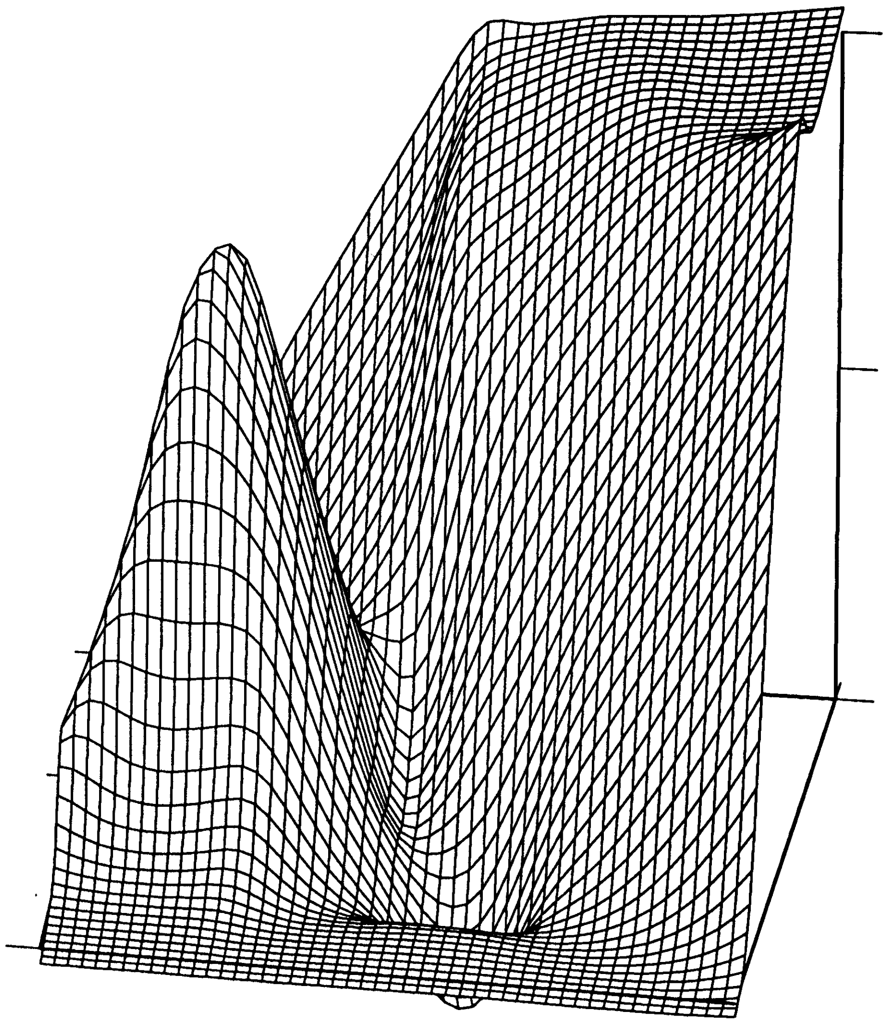
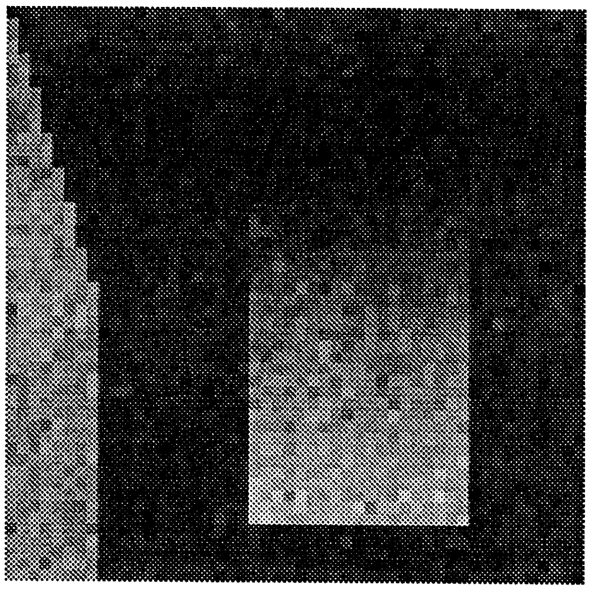


Fig (2-a)

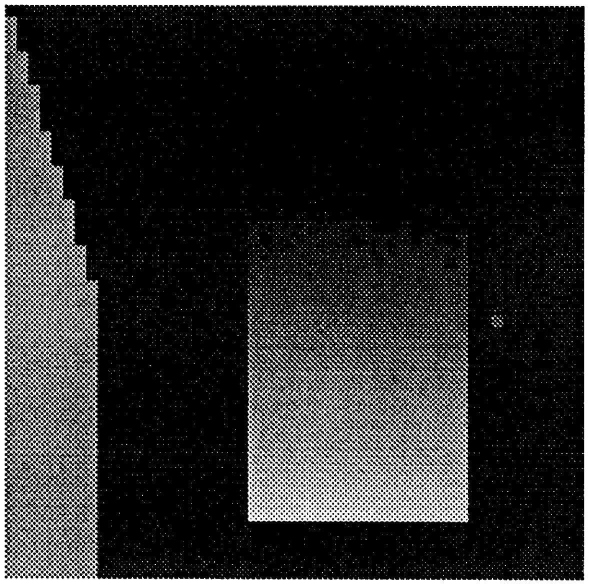

(2-6)



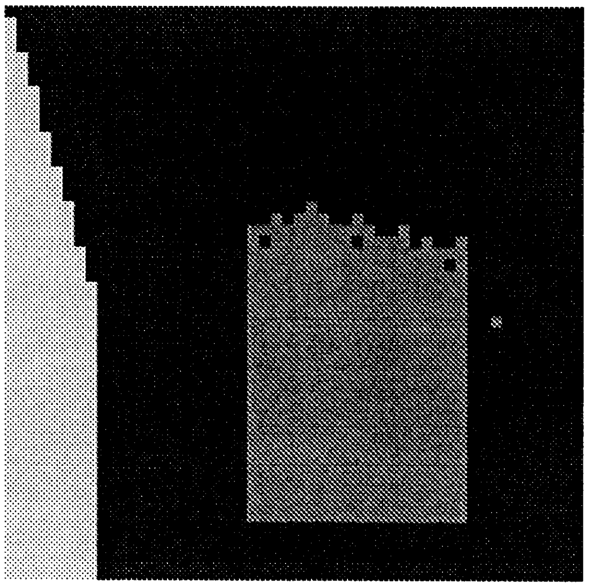
(2-c)



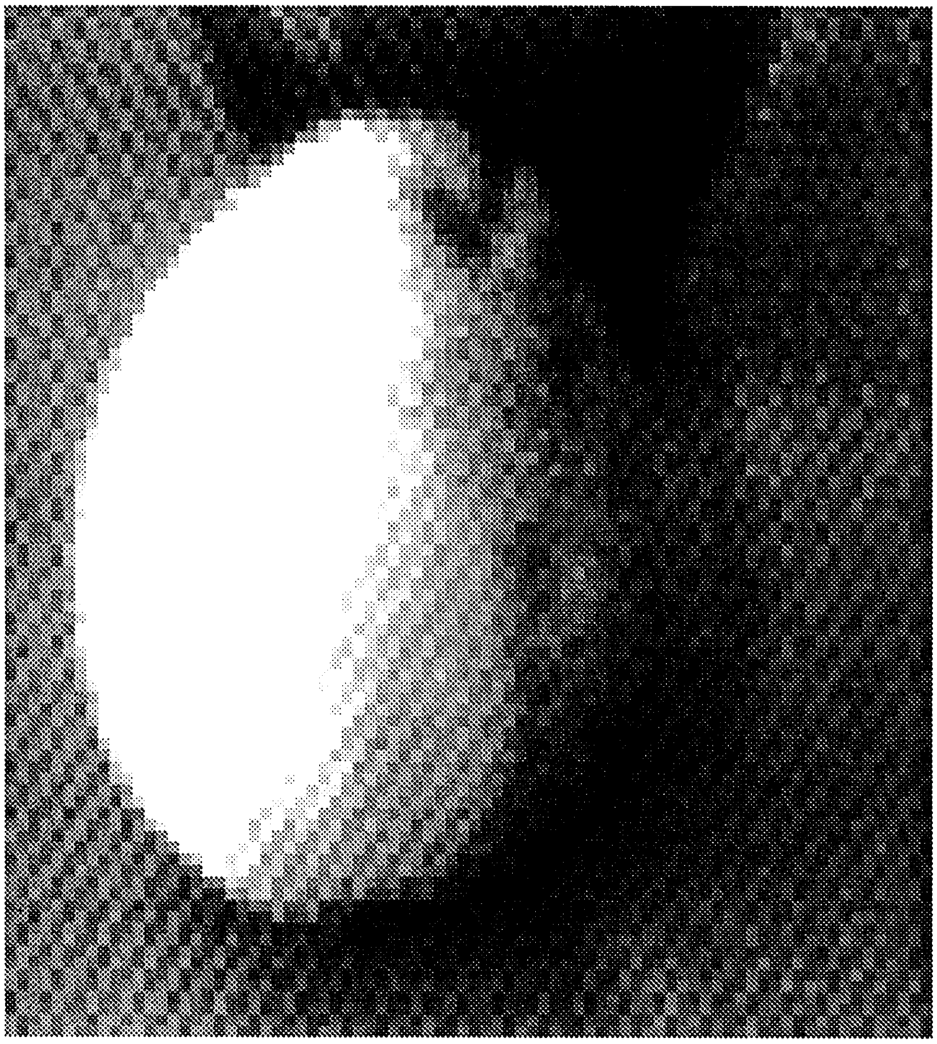
(3-a)



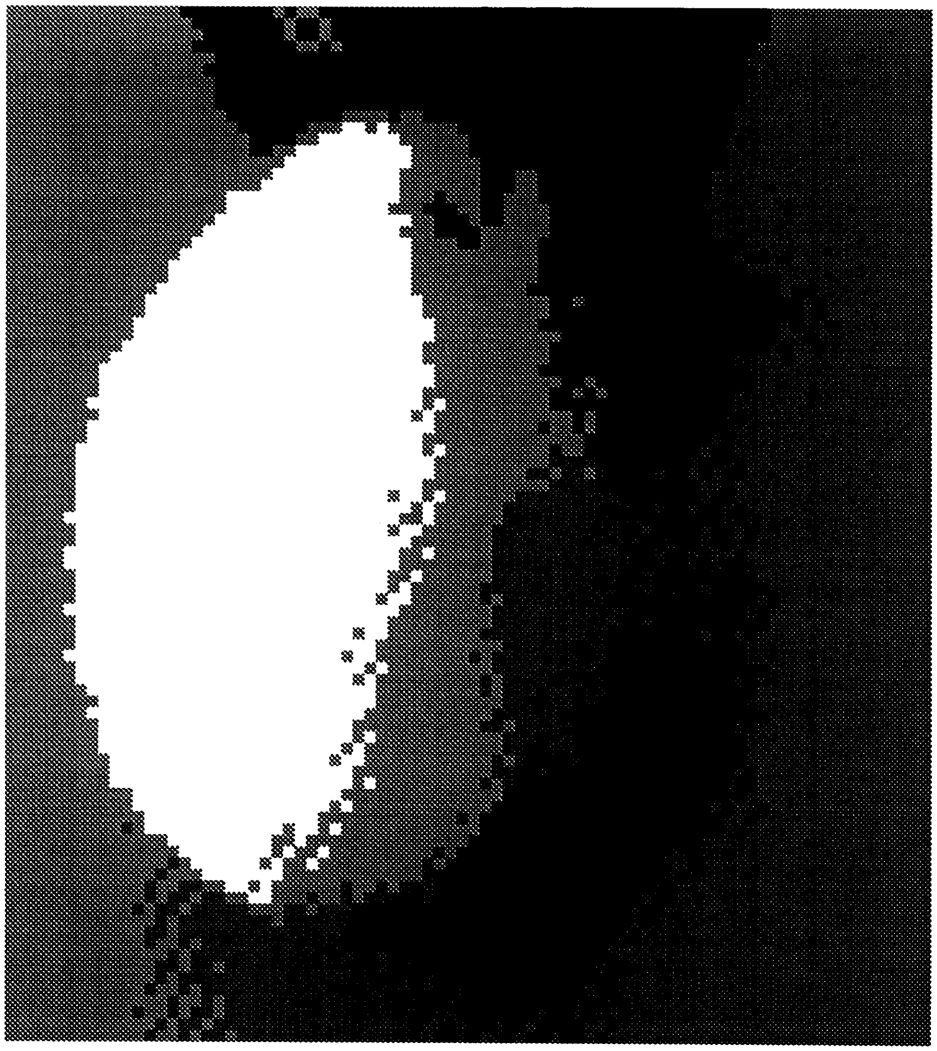
(3-b)



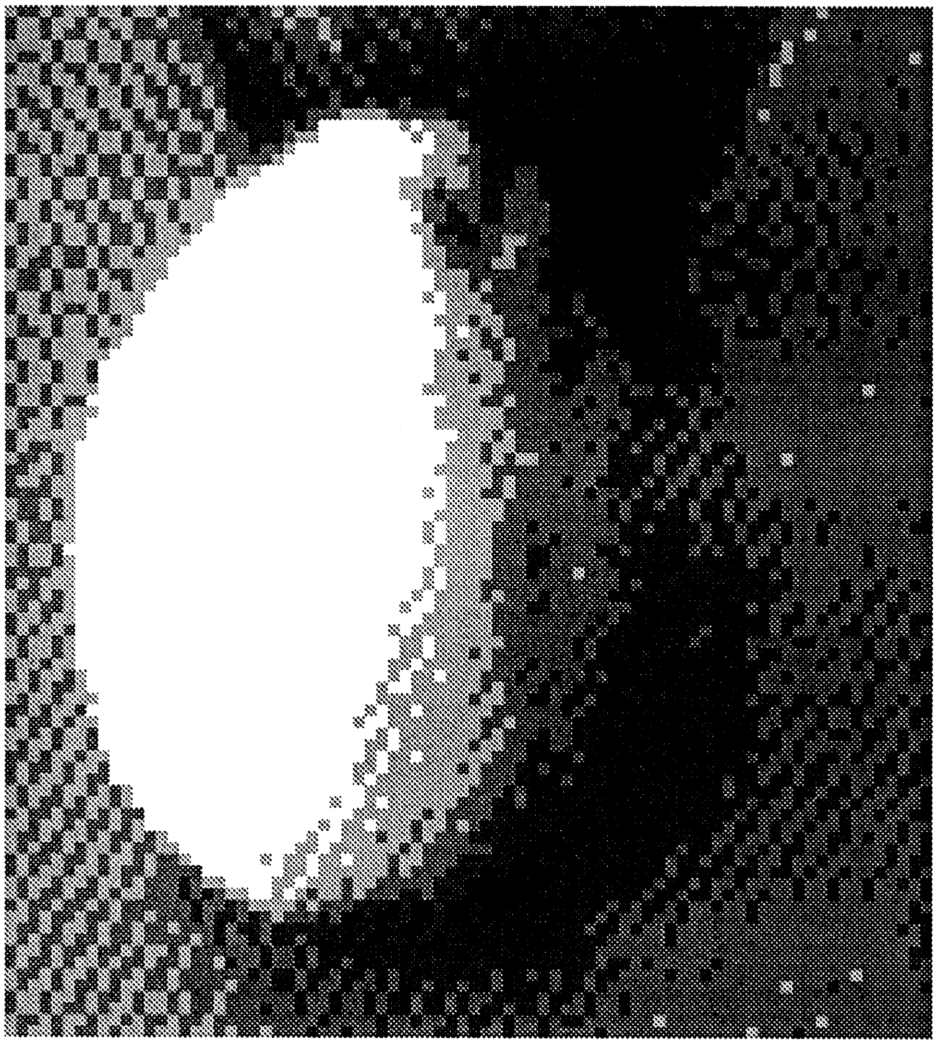
(3-c)



(4-a)



(4-6)



(4-c)