

# 2

## Coarse-to-Fine Classification and Scene Labeling

Donald Geman<sup>1</sup>

### 2.1 Introduction

The semantic interpretation of natural scenes, so effortless for humans, is perhaps the main challenge of artificial vision, having largely resisted any satisfying solution, at least in searching for multiple objects in real, cluttered scenes with arbitrary illumination. This problem is the motivation for the work in this chapter. Specifically, the models and algorithms presented here result from making computational efficiency the organizing principle for vision, a proposal recently explored in both theory [8, 2] and practice [1, 3, 4]; see also [9] and [6] for related examples of efficient visual search. “Theory” refers to analyzing the efficiency of coarse-to-fine (CTF) search under various statistical models and cost structures; summarized here are the general mathematical framework, including an abstract formulation of CTF classification based on multiresolution “tests,” and some results about optimal testing strategies. “Practice” refers to designing computer algorithms for detecting objects in natural scenes; several such experiments on face detection are included together with a brief description of how the tests are realized as image functionals.

We model scene interpretation as a dynamic and adaptive process, generating a sequence of increasingly precise interpretations. At the beginning the labels are crude and too plentiful; there are confusions among objects of interest and between objects and clutter. Eventually, the labels become more precise, for instance object categories and presentations are refined, and confusions are removed. Certain fundamental tradeoffs then evolve – between invariance and discrimination, and between false positive error and computation. Similar themes are explored in [10] for visual processing in cortex.

---

<sup>1</sup>Donald Geman is Professor, Department of Mathematics and Statistics, University of Massachusetts at Amherst.



Figure 2.1. An example illustrating multifont optical character recognition where the objective is to identify the main symbols on the license plate.

For practical convenience, we separate the whole process of scene classification into two rough phases: non-contextual and contextual. (This distinction was previously explored in [1].) Noncontextual classification, or simply *detection*, comes first. The goal is to infer from the image data instances of highly visible objects of interest under the constraint that no objects be overlooked (no “missed detections”), but allowing for a limited number of false positives. *Detection is the focus in this chapter*. As conceived here, it is based on sparse representations and performed with very simple operations – essentially just counting local features. The result is a list of “classes” of objects and their “presentations” in the scene. The desired level of detail is application-dependent; for example, the class might be “face” rather than a specific individual and the presentation might be no more specific than a range of values for certain pose parameters (e.g., position, scale and orientation). Other aspects of the presentation might be of interest, such as the font of a character or the gender of a face.

An example of multifont optical character recognition is shown in Figure 2.1 where the objective is to identify the main symbols on the license plate. The detection phase is illustrated in Figure 2.2. For each of the six characters, there are multiple detections; some *but not all* of the detections “near” each symbol are erroneous, due to clutter or confusions. There are also false alarms away from the symbols of interest. This is ongoing work with Yali Amit and will be described elsewhere.



Figure 2.2. Characters detected during detection.

Contextual classification, not treated here, involves more intensive computation in the vicinity of detections in order to determine which of these are in fact objects of interest and to disambiguate among confusions, such as D's, 0's and O's detected at roughly the same location. The underlying process is again coarse-to-fine. Moreover, ultimately there is no way to avoid a fully contextual analysis in order to discover partially visible objects and other complex spatial arrangements. Processing which accounts for context and relationships is likely to require dense representations and be computationally intensive (e.g., involve online functional optimization). One proposal is “compositional vision” (cf. [7]).

If scene labeling is driven by computational efficiency, a natural and effective mechanism is CTF classification. It is certainly *one way* of gaining (online) efficiency and I would argue that any other way ultimately boils down to something similar. CTF classification depends on a CTF *representation* for the family of interpretations under investigation. In other words, the representation of objects and presentations must be structured to accommodate coarse-to-fine search. Thus, the events of interest must be characterized by “attributes” at many levels of resolution. CTF search then means investigating those attributes in a particular order, namely from coarse ones to fine ones. This is the way we play *Twenty Questions*.

We develop an abstract formulation of CTF classification. Roughly speaking, we consider a series of nested (hence increasingly fine) partitions of the set of possible explanations, and we define a binary “test” for each cell  $\Lambda$  of each partition. The test  $X_\Lambda$  associated with  $\Lambda$  must always respond “yes” to interpretations in  $\Lambda$ . The tests also have varying levels of “cost” and “discrimination” (statistical power); both increase as cell size decreases, and hence there is a tradeoff with invariance. The “detector”  $\hat{Y}$  is a (set-valued) function of these tests; it consists of all interpretations which are confirmed at all levels of resolution, and is the primary object of our mathematical analysis. More specifically, we ask: *Which sequential (test by test) adaptive evaluation of  $\hat{Y}$  minimizes average computation?* The an-

swer is that under wide-ranging assumptions on the statistical distribution of the tests and how cost is measured, and among all testing strategies based on performing tests one at a time until a decision is reached (i.e.,  $\hat{Y}$  is determined), CTF questioning minimizes the *mean* of the sum of the costs of the all the tests which are utilized.

Further remarks about invariance and discrimination, and about parallel vs. serial processing, follow in Section 2.2. The abstract formulation is given in Section 2.3, where the statistical framework is laid out, including the definitions of cost, invariance and discrimination, and the definition of  $\hat{Y}$ . In Section 2.4 we introduce the family of possible evaluations of the detector and a model for measuring the computational efficiency of each candidate. Several results on optimal strategies are mentioned in Section 2.5 without proof and the error rates of the detector are specified in Section 2.6. In Section 2.7, we return to the scene interpretation problem and put everything in concrete terms, including how  $\hat{Y}$  is constructed from image intensity data. Finally, some experiments on face detection and concluding remarks appear in Section 2.8.

## 2.2 Invariance vs. Discrimination

The rationale for CTF search is intuitive and transparent. Start with properties of objects and presentations which are simple and common, almost regardless of discriminating power; in other words, look for tests which *invariably* accept as many object/pose pairings as possible, even if many instances of clutter and non-targeted objects are found as well. Rejecting even a small percentage of background instances with cheap and universal tests is efficient. Then proceed to more discriminating properties, albeit more complex and specialized; whereas a greater number of tests must be designed or learned in order to “cover” all objects and poses, only relatively few of them will be needed during any given search due to pruning by coarser tests. Also the still significant false alarm rate is compensated by invariance (no missed detections) and low cost termination of the search. Finally, reserve computationally intensive, highly discriminating filters (basically, object-specific and pose-specific “template-matching”) for the very end – for those inevitable and diabolical arrangements of clutter which “look” like objects in the eyes of the features.

This program amounts to creating “invariants” at many levels of power. But these are *not* the geometric and algebraic types sought after in continuum, shape-based approaches to object recognition. The invariants here are based on generic local features, not special points on curves, etc. And our requirements are more modest: Find binary image functionals which always respond positively for a given set of shapes but may respond positively to other shapes and image structures. It is only at the level of low

invariance (specific poses) that we demand high discrimination. Consequently, during the course of processing there is then a steady progression from high invariance to low invariance and from low discrimination to high discrimination.

The image functionals we consider in Section 2.7 are of the form

$$X = \begin{cases} 1 & \text{if } \sum_{l \in L} \xi_l \geq t \\ 0 & \text{otherwise} \end{cases}$$

where each  $\xi_l$  is a local binary feature which signals an “edge” is present “near” location  $z_l$  and with orientation  $\phi_l$ ;  $L$  is a distinguished set of edges dedicated to a set of poses and  $t$  is an appropriate threshold. (How “near” depends on the desired level of invariance; see Section 2.7.) Thus, evaluating  $X$  consists of checking for at least  $t$  edges among a special ensemble which characterizes a particular set of shapes – certain types of objects at certain geometric poses. The complexity of such a test might simply be  $|L|$ . High discrimination and high complexity corresponds to “template-matching” and the set  $L$  might then provide a rather dense representation of the shapes. However, for such elementary tests, achieving high invariance (covering many poses) *and* high discrimination at the same time is likely to be impossible, regardless of cost.

It is clearly impossible to find common but localized attributes of two object presentations with significantly different (geometric) poses, say far apart in the scene. As a result, we use a simple, “divide-and-conquer” strategy based on object location. (Every object is assumed to have a visible, distinguished point.) One “base” detector,  $\hat{Y}$ , is designed to find all instances of objects with presentations in a “reference” cell, for example locations confined to a  $k \times k$  block and scale confined to a  $[\sigma_{min}, 2\sigma_{min}]$  where  $\sigma_{min}$  is the smallest scale entertained. The scene is partitioned into non-overlapping  $k \times k$  blocks, and the detector  $\hat{Y}$  is applied to the image data  $I(z)$ ,  $z \in W$  in a window  $W$  centered at each such block; the dimension of  $W$  is sufficiently large to capture all objects at the given locations and scales. Objects at scales larger than  $2\sigma_{min}$  are detected by repeatedly downsampling and parsing the scene in the same way.

In principle, the detector could be applied to each window simultaneously; this is the parallel component of the algorithm. The serial component – the CTF implementation of  $\hat{Y}$  in each window – is the heart of the algorithm and the real source of efficient computation.

### 2.3 CTF Classification: Abstract Formulation

Let  $\mathcal{S} = \{\lambda_1, \lambda_2, \dots\}$  denote a set of *states* or *interpretations*. Each subset  $\Lambda \subset \mathcal{S}$  will be called an *index*. In addition, fix a probability space  $(\mathcal{I}, P)$  and suppose there is a *true index*  $Y(I)$  for each  $I \in \mathcal{I}$ . Although we allow

more than one true interpretation, we are primarily interested in the case in which either  $Y = \{\lambda\}$  or  $Y = \emptyset$ .

In the application to detecting objects,  $\lambda$  is a pair  $(c, \theta)$  where  $c$  is the “class” of an object and  $\theta$  stands for the “presentation.” Even one object class is challenging and frequently considered in computer vision.  $\mathcal{I}$  is then the set of subimages  $I = \{I(z), z \in W\}$  and  $P$  could be taken as an empirical measure; we shall be more specific about this later on. Finally,  $Y(I)$  is the list of the objects and presentations appearing in  $I$ . In general, there is at most one object which is both visible and centered in a given  $W$ .

An important feature of the detection problem, and one that motivates an upcoming approximation of  $P$ , is that  $P(Y = \emptyset) \gg P(Y = \Lambda)$  for any given  $\Lambda$ . We might even assume that  $P(Y = \emptyset) \gg P(Y \neq \emptyset)$ , so that the most likely interpretation is that there are no states “present” in  $I$ . Write  $P_\lambda$  for  $P(\cdot | \lambda \in Y)$  and  $P_0$  for  $P(\cdot | Y = \emptyset)$ .

Shortly we shall define a *detector*  $\hat{Y}$  based on a family of functions  $X : \mathcal{I} \mapsto \{0, 1\}$  called *tests*. The basic constraint on  $\hat{Y}$  is zero false negative error:

$$P(Y \subset \hat{Y}) = 1 \quad (2.1)$$

Equivalently,

$$P_\lambda(\lambda \in \hat{Y}) = 1, \quad \forall \lambda \in \mathcal{S}.$$

Assume each test has a *cost* or *complexity*  $c(X)$  which represents the amount of online computation (or time) necessary to evaluate  $X$  and of course depends on how  $X$  is constructed. The *invariant set* for  $X$  is  $\Lambda(X) = \{\lambda : P_\lambda(X = 1) = 1\}$ . Finally, the *discrimination* or *power* of  $X$  is defined as  $\beta(X) = P_0(X = 0)$ . The tradeoff between cost and discrimination at different levels of invariance is shown in the lefthand panel of Figure 2.3; the righthand panel shows the tradeoff between invariance and discrimination at different costs.

Suppose we are given a family of tests  $\mathbf{X} = \{X_\Lambda, \Lambda \in \mathbf{\Lambda}\}$  where the notation  $X_\Lambda$  means that  $\Lambda = \Lambda(X)$ . The reason for indexing the tests by their invariant sets is that we will build tests to a set of specifications. Basically, we *first* design a hierarchy of subsets of  $\mathcal{S}$  and *then*, for each  $\Lambda$  in the hierarchy, we build a test  $X_\Lambda$  which is invariant with respect to the classes and poses in  $\Lambda$ .

Now define  $\hat{Y}(I) \subset \mathcal{S}, I \in \mathcal{I}$ , by

$$\hat{Y}(I) = \hat{Y}(\mathbf{X}(I)) = \{\lambda : X_\Lambda(I) = 1 \quad \forall \lambda \in \Lambda\}. \quad (2.2)$$

The rationale is that we accept a state  $\lambda$  as part of our interpretation if and only if this state it is “verified” at all levels of resolution, in the sense that each test  $X$  which “covers”  $\lambda$  (meaning  $\lambda \in \Lambda(X)$ ) responds positively.

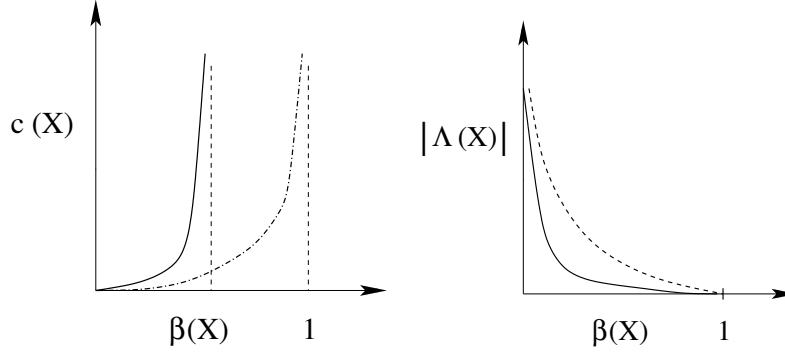


Figure 2.3. Left: The cost vs. discrimination tradeoff at two levels of invariance, “high” (solid line) and “low” (dashed line). Right: The invariance vs. discrimination tradeoff at two levels of cost, “low” (solid line) and “high” (dashed line).

## 2.4 Computation

Consider now adaptive (sequential) evaluations of  $\hat{Y}$ , i.e., tree-structured representations of  $\hat{Y}$ . Let  $\mathcal{T}$  be the family of such evaluations. Each internal node of  $T \in \mathcal{T}$  is labeled by a test  $X_\Lambda$  and each external node is labeled by an index – a subset of states. Our goal is to find the  $T$  which minimizes the mean cost of determining  $\hat{Y}$  under assumptions on how  $c(X)$  varies with  $\beta(X)$ , and how  $\mathbf{X}$  is distributed under the “background model”  $P_0$ .

To illustrate such a computational procedure, take a simple example with  $\mathcal{S} = \{\lambda_1, \lambda_2\}$  and three tests corresponding to  $\Lambda_{1,2} = \{\lambda_1, \lambda_2\}$ ,  $\Lambda_1 = \{\lambda_1\}$ ,  $\Lambda_2 = \{\lambda_2\}$ . One evaluation of  $\hat{Y}$  is shown in Figure 2.4 where branching left means  $X = 0$  and branching right means  $X = 1$ .

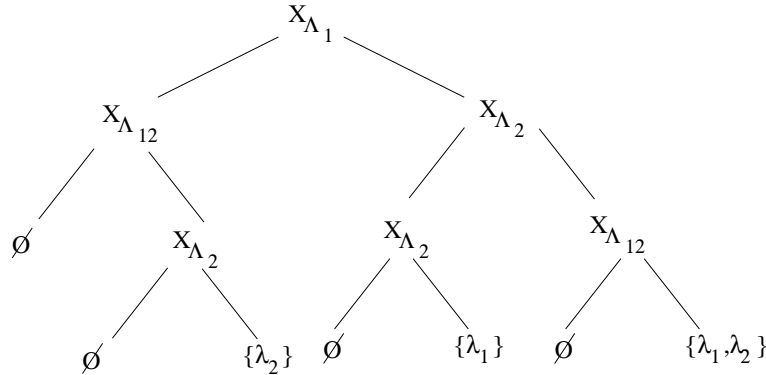


Figure 2.4. An example of a tree-structured evaluation of the detector  $\hat{Y}$

Notice that  $\hat{Y} = \emptyset$  if and only if there is a *null covering* of  $\mathbf{\Lambda}$ : a subfamily  $\{\Lambda_i\}$  such that  $\bigcup_i \Lambda_i = \mathcal{S}$  and  $X_{\Lambda_i} = 0$  for each  $i$ . In Figure 2.4, one null covering corresponds to  $\{X_{\Lambda_1} = 0, X_{\Lambda_2} = 0\}$  and one to  $\{X_{\Lambda_{1,2}} = 0\}$ .

#### 2.4.1 Mean Cost

The *cost* of  $T$  is defined as

$$C(T) = \sum_{r \in T^\circ} \mathbf{1}_{H_r} c(X_r)$$

where  $T^\circ$  is the set of internal nodes of  $T$ ,  $X_r$  is the test at node  $r$  and  $H_r$  is the history of node  $r$  – the sequence of test results leading to  $r$ . (Equivalently,  $C(T)$  is the aggregated cost of reaching the (unique) terminal node in  $T$  determined by  $\mathbf{X}$ .) Notice that  $C(T)$  is a random variable. The *mean cost*  $EC(T)$  is then

$$EC(T) = \sum_{r \in T^\circ} c(X_r) P(H_r) \quad (2.3)$$

$$= \sum_X c(X) P(X \text{ performed in } T) \quad (2.4)$$

The second expression (2.4) is useful in proving the results mentioned in the following section.

Our optimization problem is then

$$\min_{T \in \mathcal{T}} EC(T) \quad (2.5)$$

In other words, find the best testing strategy. As it turns out, the best strategies are often far more efficient than  $T$ 's constructed with top-down, greedy procedures, such as those utilized in machine learning for building decision trees; see [5] for comparisons.

#### 2.4.2 Hierarchical Tests

In order to rank computational strategies we must impose some structure on both  $\mathbf{\Lambda}$  and the law of  $\mathbf{X}$ . From here on we consider the case of *nested binary partitions* of  $\mathcal{S}$ :

$$\mathbf{\Lambda} = \{\Lambda_{m,j}, j = 1, \dots, 2^m, m = 0, 1, \dots, M\}.$$

Thus,  $\Lambda_{0,1} = \Lambda_{1,1} \cup \Lambda_{1,2}$ ,  $\Lambda_{1,1} = \Lambda_{2,1} \cup \Lambda_{2,2}$ , etc. In Section 2.7, in experiments with a single object class, the hierarchy  $\mathbf{\Lambda}$  is a “pose decomposition” wherein “cells” at level  $m+1$  represent more constrained poses than cells at level  $m$ . As  $m$  increases, the level of invariance decreases and, in the models below, the level of discrimination increases. *This tree-structured hierarchy should not be confused with a tree-structured evaluation  $T$  of  $\hat{Y}$ .* The label  $\hat{Y}$  at a terminal node of  $T$  depends on  $\mathbf{X}$  and consists of all states in *any*



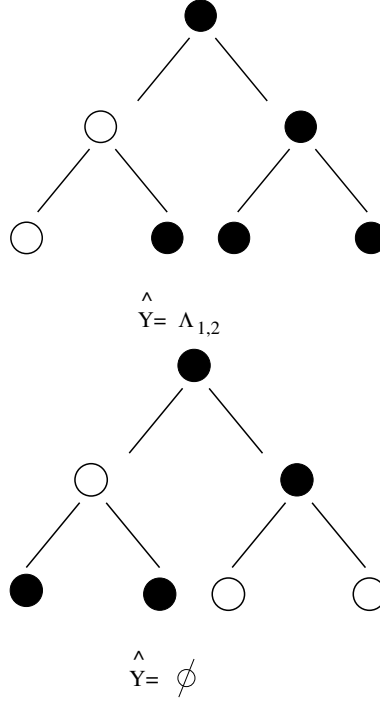


Figure 2.5. Examples of  $\hat{Y}$  for two realizations of the tests in a nested hierarchy with three levels. Dark circles represent  $X = 1$  and open circles represent  $X = 0$ . Top: There are two “chains of ones”. Bottom: There is a null covering.

level  $M$  cell of  $\Lambda$  for which there is a “1-chain” back to  $\Lambda_{0,1}$ , i.e.,  $X_{\Lambda'} = 1$  for every  $\Lambda' \supset \Lambda$ .

As a simple illustration, consider the case  $M = 2$ , in which case there are exactly seven sets in the hierarchy. Notice that the most refined cells  $\Lambda_{2,j}$ ,  $j = 1, 2, 3, 4$ , may each contain numerous states  $\lambda$ , i.e., may provide an interpretation at a level of resolution which is still rather “coarse.” Figure 2.5 shows  $\hat{Y}(\mathbf{X})$  for two realizations of  $\mathbf{X}$ . For the same hierarchy and realizations of  $\mathbf{X}$ , the lefthand panel of Figure 2.6 illustrates a “depth-first” CTF evaluation of  $\hat{Y}$  and the righthand panel a “breadth-first” CTF evaluation.

Another way to interpret  $\hat{Y}$  is the following: For each level  $m$  in the hierarchy define

$$\hat{Y}_m(\mathbf{X}) = \bigcup_{j \in J_m(\mathbf{X})} \Lambda_{m,j}$$

where  $J_m = \{1 \leq j \leq 2^m : X_{m,j} = 1\}$ . The detector  $\hat{Y}$  is  $\bigcap_m \hat{Y}_m$ . We can think of  $\hat{Y}_m$  as an estimate of  $Y$  at “resolution”  $m$ . Necessarily,  $Y \subset \hat{Y}_m$

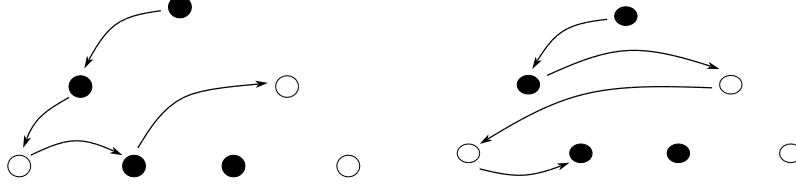


Figure 2.6. As in Figure 2.5, dark and light circles indicate positive and negative tests for a three level hierarchy. Left: A “depth-first” coarse-to-fine search. Right: A “breadth-first” coarse-to-fine search.

for each  $m$ , although in general it needn’t happen that  $\hat{Y}_{m+1} \subset \hat{Y}_m$  (or vice-versa).

### 2.4.3 An Approximation

We are going to assume that the mean cost is computed with respect to the background distribution  $P_0$ . This is motivated by the following argument. Recall that labeling the entire scene means applying the detector  $\hat{Y}$  to subimages  $I$  centered on a sparse sublattice of the original scene. Whereas the likelihood of having some objects in the entire scene may be large, the *a priori* probability of the “null hypothesis”  $Y = \emptyset$  is approximately one when evaluating  $\hat{Y}$  for an arbitrary subimage at the scale of the objects. Therefore, the average amount of computation involved in executing our detection algorithm with a particular strategy  $T$  can be approximated by taking the expectation of  $C(T)$  under  $P_0$ . Of course this approximation degrades along branches with many positive test responses, especially to discriminating tests associated with “small” class/pose cells, in which case the probability of an object being present may no longer be negligible. If one were to measure the mean computation under an appropriate (mixture) distribution, the conditional distribution of the tests under the various object hypotheses would come into play. Nonetheless, we continue to compute the likelihood of events under  $P_0$  alone, thereby avoiding the need to model the behavior of the tests given objects are present.

## 2.5 Optimality Results

For simplicity, write  $X_{m,j}$  for  $X_{\Lambda_{m,j}}$ . From here on, we make the following assumptions about the the distribution of  $\mathbf{X}$ :

- $\{X_{m,j}\}$  are independent random variables under  $P_0$ ;

- $\beta_{m,j} \doteq P_0(X_{m,j} = 0) = \beta_m$ ,  $m = 0, 1, \dots, M$ ;
- $\beta_0 \leq \beta_1 \leq \dots \leq \beta_M$ ;
- $c(X_{m,j}) = c_m$ ,  $m = 0, 1, \dots, M$ ;
- $c_m = \Phi(\beta_m)$  with  $\Phi(0) = 0$  and  $\Phi$  increasing.

The independence assumption is violated in practice, but we make it in order to facilitate a theoretical analysis. The other assumptions are realistic, partly by design, although the power of the tests may differ slightly within levels.

### 2.5.1 Testing $\hat{Y} = \emptyset$ vs. $\hat{Y} \neq \emptyset$

Consider first the problem of determining whether or not  $\hat{Y} = \emptyset$ , in other words, evaluating  $\hat{Z} = \mathbf{1}_{\{\hat{Y} \neq \emptyset\}}$ . The set-up is the same, except the terminal labels of  $T$  are simply “0” or “1.” This problem was studied in [3] and [8] under the assumption that  $\Phi$  is *convex*, i.e., cost is a convex, increasing function of power. One strategy is the depth-first CTF strategy, illustrated in Figure 2.7 for the case  $M = 2$ .

In Figure 2.8 two particular sample paths (branches) are depicted from a depth-first, CTF search of a five level hierarchy. The path on the left leads to the label  $\hat{Z} = 0$  and the one on the right leads to  $\hat{Z} = 1$ .

The following result is proved in [8]; earlier, [3] had shown that the coarsest test ( $X_{0,1}$ ) is necessarily at the root. The convexity assumption can be relaxed to supposing that  $\frac{c_m}{\beta_m}$  is increasing,  $m = 0, \dots, M$ .

**Theorem:** *If  $\Phi$  is convex, and  $P = P_0$ , depth-first CTF search is the optimal strategy for evaluating  $\hat{Z}$ .*

### 2.5.2 Determining $\hat{Y}$

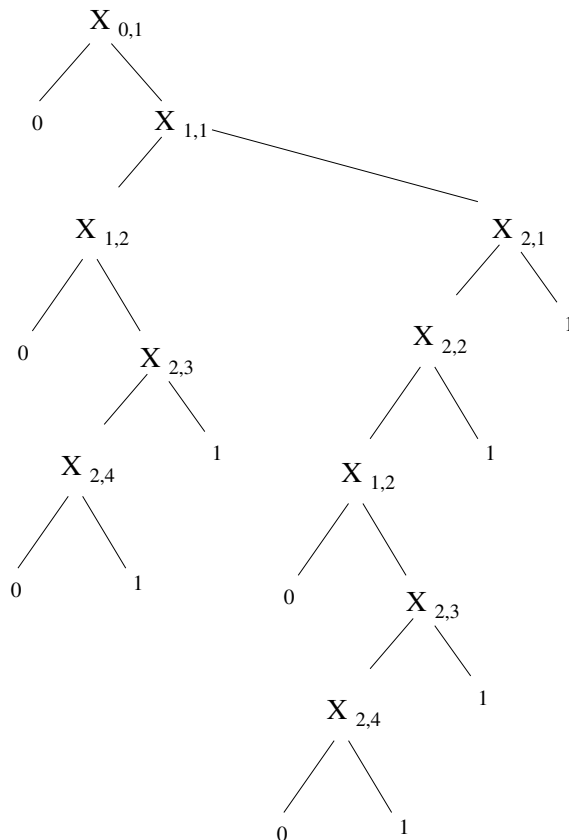
Here the objective is to determine *all* 1-chains instead of merely whether or not one such chain exists. For either CTF strategy, the expected cost under  $P_0$  is

$$E_0 C(T) = c_0 + \sum_{m=1}^M c_m 2^m \prod_{j=0}^{m-1} (1 - \beta_j).$$

As it turns out, this is the smallest possible mean cost:

**Theorem:** *The CTF strategy is optimal for any increasing cost sequence if  $\beta_0 > .5$*

The proof of this result, and others based on varying cost models and test hierarchies, will appear in [2].

Figure 2.7. The CTF strategy tree for the case  $M = 2$ .

## 2.6 Error

We briefly discuss the theoretical error rates of the detector  $\hat{Y}$  defined in (2.2).

In principle, the false negative rate is null. (Of course, in practice, this requires that  $X_\Lambda$  be invariant for  $\Lambda$ , which can be difficult to achieve.) The false positive error

$$\delta(\hat{Y}) \doteq P_0(\hat{Y} \neq \emptyset)$$

is determined by the (joint) distribution of  $\{X_{m,j}\}$  under  $P_0$ , and depends on  $\beta_0, \dots, \beta_M$ . The rate *per pixel* is then  $\frac{\delta(\hat{Y})}{k^2}$ ; recall that the search for object locations is conducted in non-overlapping  $k \times k$  blocks. A crude bound is to replace the probability of at least one 1-chain under  $P_0$  by the

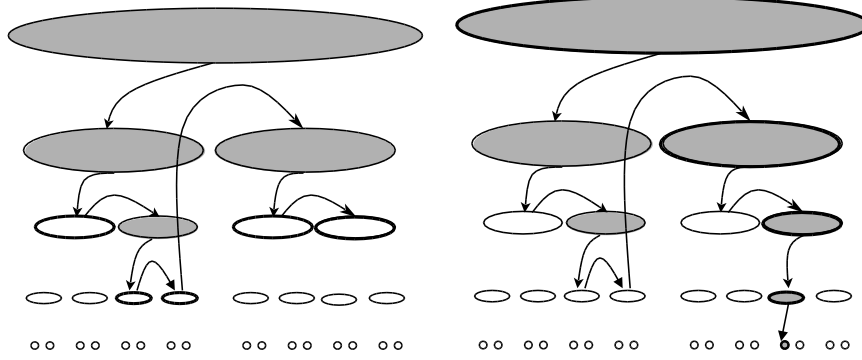


Figure 2.8. Two branches from a depth-first CTF search for a hierarchy with five levels. The tests are explored in the indicated order, with gray indicating a positive answer and white in bold outline indicating a negative answer; the other tests were not evaluated. Left: A null covering is encountered, ending the search. Right: A chain of ones is encountered (Gray in bold outline), ending the search when the goal is to determine if  $\hat{Y} = \emptyset$ .

sum of the probabilities, yielding

$$\begin{aligned} \delta(\hat{Y}) &\leq 2^M \prod_{m=0}^M (1 - \beta_m) \\ &\leq \frac{1}{2} \gamma^{M+1}, \quad \gamma = 2(1 - \beta_0). \end{aligned}$$

To calculate  $\delta(\hat{Y})$  exactly, notice that there is a correspondence between realizations of a breath-first CTF evaluation of  $\hat{Y}$  and realizations of a branching process with  $M$  generations starting from a single individual. Consequently,

**Theorem:** *The false positive error  $\delta(\hat{Y})$  is the probability of no extinction for a non-homogeneous branching process with binomial family law  $\text{Bin}(2, 1 - \beta_m)$  at generation  $m = 0, \dots, M$ .*

## 2.7 Application to Face Detection

Recall that each  $\lambda \in \mathcal{S}$  corresponds to the presentation or instantiation of an “object” in a predetermined library. The presentation includes information about the position, scale and other aspects of the geometric pose, and perhaps other properties of interest.

### 2.7.1 One Generic Class

Consider the simplified scenario of one generic object class and linear pose. More specifically, consider the problem of detecting all instances of frontal views of faces. The global procedure is to parse the scene at various scales, and at a sampling of locations, with a window of size  $64 \times 64$ , in each case applying a detector which computes the list of poses present in the window. In this case, the output  $\hat{Y}$  of processing a window is simply a list of poses. What follows is a brief description of the algorithm; the details can be found in [4].

### 2.7.2 Pose Decomposition

The pose of a face is defined in the image plane, given by  $\theta = (z, \sigma, \psi)$ , where  $z$  is the center point between the eyes,  $\sigma$  is the distance (in pixels) between the eyes and  $\psi$  is the “tilt.” The image is partitioned into non-overlapping  $8 \times 8$  blocks, and the basic detector  $\hat{Y}$  is applied to the image data in a window centered at each such block. These windows are the subimages in  $\mathcal{I}$  in the previous sections. The detector  $\hat{Y}$  produces a list of poses with  $z \in [28, 36]^2$ ,  $8 \leq \sigma \leq 16$ ,  $-20^\circ \leq \psi \leq 20^\circ$ ; of course in this case these are either false alarms or responses to the same face. Call this set of poses the “reference cell”  $\Lambda_{0,1}$ . Faces of scale  $16 \leq \sigma \leq 32$  are found by downsampling the original scene and parsing again, etc.

The hierarchy  $\{\Lambda_{m,j}\}$  of subsets of (reference) poses is constructed by recursively partitioning  $\Lambda_{0,1}$  into a sequence of nested partitions; each cell  $\Lambda$  of each partition is a subset of poses which is included in exactly one of the cells in the preceding, coarser partition. At each level  $m = 1, 2, \dots, M$ , one component of the pose  $\theta$  is subdivided into equal parts – binary splits for scale and tilt and quaternary splits for position. Thus, for example, each cell  $\Lambda_{1,j}$ ,  $j = 1, 2, 3, 4$ , of the the first partition corresponds constraining  $z$  to one of the four  $4 \times 4$  subsquares of of the initial  $8 \times 8$  square. There are two splits on  $z$ , two on  $\sigma$  and two on  $\psi$ , resulting in  $M = 6$  levels (excluding the root).

### 2.7.3 Learning

Each test  $X_{m,j} = X_{\Lambda_{m,j}}$  checks for a certain number of distinguished edge fragments, and hence is defined by a list  $L_{m,j}$  of edges and a threshold (as described in Section 2.2) both determined during training. Consequently, the tests  $X$  are simply counting operators. Checking for an edge means evaluating a binary local feature  $\xi_l$  indexed by a position  $z_l$  and an orientation  $\phi_l$  as described in Section 2.2. However, there is another, crucial, parameter – the “tolerance” of the edge – which allows one to achieve invariance to the poses in  $\Lambda$ . (The MAX filter in [10] has the same aim.) The tolerance  $\eta$  is the length of a strip of pixels centered at  $z_l$  and perpendicular

to the direction of the edge. The feature  $\xi_l = 1$  if there is an edge at the given orientation at *any* location in the strip. Thus,  $\eta$  controls the amount of ORing, which in turn depends on the desired degree of invariance. All  $\xi$  in the list have the same  $\eta$ . The tolerance  $\eta$  is “large” for the coarse cells  $\Lambda_{m,j}$  and “small” for the fine cells. It controls the tradeoff between invariance and discrimination.

All the tests  $X_{m,j}$  in the hierarchy are built with the same learning algorithm; the lists differ due to varying training sets, corresponding to varying constraints on the set of poses. The experiments shown here are based on the ORL database which contains 400 grayscale face pictures of size  $112 \times 96$  pixels. For each cell  $\Lambda_{m,j}$ , a synthetic training set of 1600 face images is constructed whose poses are in  $\Lambda_{m,j}$ . Again, the details are in [4], including how the lists of edges  $\xi_l, l \in L$  are chosen.

## 2.8 Experiments and Conclusions

The experiments use a breadth-first CTF evaluation of  $\hat{Y}$ . Nearby detections are clustered, resulting in one estimated pose per face, indicated by a triangle. The false negative rate is not null, and there are false positives, on the order of 1 – 10 per scene on average. Thus, when computation is efficiently organized, one can use very simple components (such as the counters  $X$  described in the previous section) and still achieve reasonable error rates, in fact comparable to the best ones reported in the literature for high resolution images; see for example [11]. In addition, detection is extremely fast, well under one second for a scene on the order of  $400 \times 400$ , which is faster than previously reported results. Two results are shown in Figures 2.9 and 2.11. As seen in Figure 2.12, coarse-to-fine processing leads to highly asymmetric scene processing in terms of spatial concentration, with orders of magnitude differences in the application of resources to different regions of the scene.

Recall that detection is far from a complete solution to scene interpretation. It leaves confusions unresolved and does not address occlusion and other complicating factors. Many examples of such specific class/pose confusions can be seen in a higher resolution rendering (not shown here) of the labeled detections in Figure 2.11. However, if highly visible objects are sure to be detected with a limited number of false alarms, it may then be computationally feasible to entertain very intense processing which is optimization-based but highly localized.

Finally, for detection, we can suppose that each test is constructed during an offline training phase, as in the previous section. Indeed, since we are not anticipating the specific confusions and occlusion patterns that might arise, we can afford to make a list of *all* the tests we wish to construct, learn them during training, and store all the instructions for execution. On the



Figure 2.9. An experiment from the CNN website.



Figure 2.10. An experiment with a group photograph.

contrary, during contextual classification we might be obliged to generate hypotheses and construct tests online, i.e., during scene parsing. Such ideas are currently being explored in the context of character recognition.





Figure 2.11. The detections for the group photo.

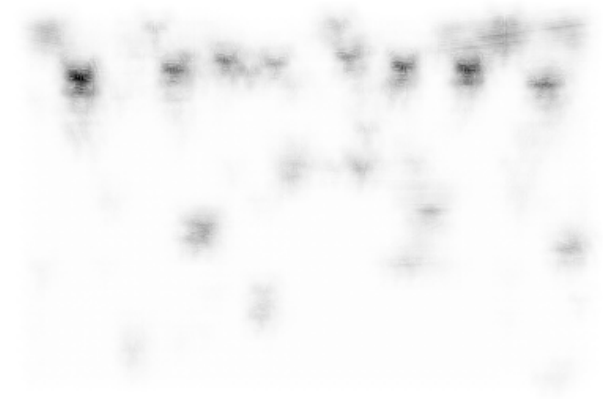


Figure 2.12. The coarse-to-fine nature of the algorithm is illustrated for the group photo by counting, for each pixel, the number of times the detector checks for the presence of an edge in its vicinity. The level of darkness is proportional to this count.

## References

- [1] Amit, Y. and D. Geman. 1999. A computational model for visual selection. *Neural Computation*, 11:1691-1715.
- [2] Blanchard, G. and D. Geman. 2002. Computational models for coarse-to-fine search, in preparation.
- [3] Fleuret, F. 2000. Detection hiérarchique de visages par apprentissage statistique. Ph.D. thesis, University of Paris VI, Jussieu, France.

- [4] Fleuret, F. and D. Geman. 2001. Coarse-to-fine face detection. *Inter. J. Computer Vision*, 41:85-107.
- [5] Geman, D. and B. Jedynak. 2001. Model-based classification trees. *IEEE Trans. Info. Theory*, 47:1075-1082.
- [6] Geman, S., K. Manbeck, and D. McClure. 1995. Coarse-to-fine search and rank-sum statistics in object recognition. Technical Report, Division of Applied Mathematics, Brown University.
- [7] Geman, S., D. Potter, and Z. Chi. 2001. Composition systems. *Quarterly of Applied Mathematics*, to appear.
- [8] Jung, F. 2001. Reconnaissance d'objets par focalisation et detection de changements. PhD thesis, Ecole Polytechnique, Paris, France.
- [9] Lambdan, Y. and J.T. Schwartz and H.J. Wolfson. 1988. Object recognition by affine invariant matching. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 335-344.
- [10] Riesenhuber, M. and T. Poggio. 1999. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019-1025.
- [11] Rowley, A.R. 1999. Neural network-based face detection. PhD Thesis, Carnegie Mellon University.

# 3

## Environmental Monitoring Using a Time Series of Satellite Images and Other Spatial Data Sets

Harri Kiiveri, Peter Caccetta,  
Norm Campbell, Fiona Evans,  
Suzanne Furby, and Jeremy Wallace<sup>1</sup>

### 3.1 Introduction

As a result of extensive farmland clearing over the last hundred years or so, dry-land salinity is a major problem in Western Australia. In fact, in some parts of the state, over 20 percent of Agricultural land is no longer productive. Prior to the work to be described in this chapter, no reliable large scale estimates of the extent or progression of salinity were available. This chapter describes a methodology for monitoring the historical extent of salinity, using a time series of satellite imagery, landform information derived from digital elevation models and ground truth data collected by experts with local knowledge. This work has served to highlight the salinity problem to decision makers in government and to provide input into the process of developing and applying remedial measures to arrest the spread of salinity.

Although the chapter primarily refers to salinity monitoring, the methodology is quite general and can be used in any situation in which the relationships amongst a number of images can be represented by a directed or undirected graph.

The chapter is structured as follows. The area monitored and the data used are described in Section 3.2. A short description of the data (pre)processing is given in Section 3.3. Whilst we don't not spend much

---

<sup>1</sup>The authors are all with the CSIRO Division of Mathematical and Information Sciences, Floreat Park, Western Australia.

time in Section 3.3, this part of the work is the most time consuming and critical to the success of the project. In Section 3.4 a model for integrating a time series of satellite images with other spatial data sets is constructed. Since the model contains latent (unobserved) images we consider the issue of identifiability of parameters in the model and also describe an EM algorithm for estimating the parameters. The model also explicitly allows for uncertainty in the inputs and outputs of the monitoring process. In Section 3.5, the chapter concludes with an example of the output products produced by the methodology.

### 3.2 Description of the data

The study area is covered by 15 Landsat TM scenes [18] and covers approximately 18 million hectares. This area, with satellite scenes overlaid, is shown in Figure 3.1. For each scene we have 6 satellite images, roughly every two years over the time period 1988 to 1999. Each satellite image has 8000 by 8000 pixels (picture elements). The images were re-sampled to give a pixel size of 25 metres and consist of six wavelength channels or bands. Hence for each date, at each pixel we have a data vector with six components. Previous work [10] has identified September/October to be the optimal time to discriminate salt from other ground classes and the images were acquired as close to this window as possible.

In addition, we have a landform map (image) derived from a digital elevation model for each scene. This is a five-class map identifying hilltops, valley floors and varying degrees of slopes.

Training data in the form of interpreted salinity maps over relatively small regions was available for each scene. These were supplied by local experts.

All in all for each scene we have approximately 4.5 gigabytes of data to store and process. Any methods we use to process this data need to take the large data volumes into consideration.

### 3.3 Data pre-processing

To produce input data for the model to be discussed in the next section required a large amount of data processing. Firstly, contour data was interpolated to produce grided digital elevation models. These were processed to produce water accumulation maps which were then converted into Landform maps, see [5].

Secondly, for each scene, the time series of satellite images needed to be accurately geo-coded or rectified. This was done carefully to ensure that



Figure 3.1. Location of satellite scenes over the study area

location errors were less than the pixel size [7]. This is essential to ensure that changes in ground cover are not confounded with locational errors.

Thirdly, images were calibrated to like values to ensure that areas which are not changing over time look the same in each image. To do this, linear transformations were defined using robust regression techniques to make the values of the images over a common set of (pseudo) invariant targets as close as possible [11].

Finally, each image was classified into approximately 12 ground cover classes, e.g., bush, water, agriculture, bare ground and different types of salt. The method used was Gaussian maximum likelihood [19] and this required the careful and time-consuming selection of training sites within each image. Using canonical variate analysis [6] the training site separation was studied and the training sites were eventually amalgamated into the final ground cover training classes. Figure 3.2 shows an idealised canonical variate plot and also illustrates the difficulty in using the satellite imagery to map salinity. The circles/ellipses in Figure 3.2 show the locations in canonical variate space of the cover types of interest. Unfortunately, there is an over lap with salt and bare, salt and bush and salt and agriculture

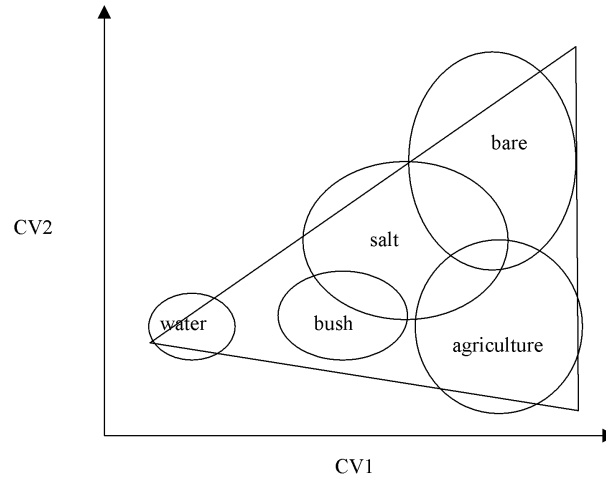


Figure 3.2. Idealised canonical variate plot showing ground cover separation

which indicated that the satellite imagery on its own could not discriminate between salt and other significant ground cover types.

Note that for each scene, the image for each date was classified i.e. 6 classifications were done.

Previous work had shown that the Gaussian assumption was reasonable for this problem. Although we used the Gaussian maximum likelihood classifier, any method which produces probabilities of class membership and has a notion of typicality could have been used.

### 3.4 A conditional probability network for image data

In the previous section we mentioned the difficulty in mapping salinity given a satellite image on one single date as the satellite sensor is unable to distinguish between salt and certain other ground cover classes. However, some simple prior knowledge about the process of salinisation suggests that a time series of images in conjunction with landform information can be used to effectively map salinity. It is known that salt is stable over time, so for example, paddocks which appear salt affected but in fact are bare, can be correctly mapped by noticing the temporal cover class sequence. If apparent salt is followed by healthy crop cover then we can deduce that the paddock is not salt affected. Similarly we know that salt tends to occur mainly in valley floors so that apparent salt in hill tops or upland regions

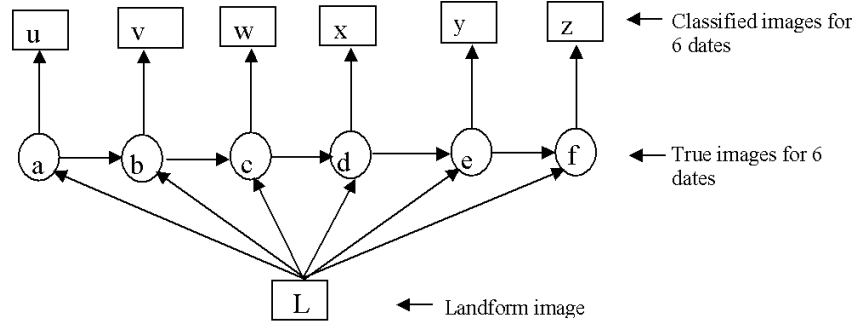


Figure 3.3. A conditional probability network (CPN) to integrate all the data for an area.

can also be down weighted. In this section we will develop a model for each scene which enables us to take this type of prior information into account.

#### 3.4.1 The model

Conditional probability networks (CPNs) are typically represented by a directed acyclic graph, where the nodes represent variables and the edges define parent child relationships amongst the variables. Associated with the graph is a factorisation of the joint probability distribution of all the variables [12]. If we allow the nodes of the graph to represent images, then a model for a time series of images could be represented as follows in Figure 3.3.

In Figure 3.3, the six classified satellite images are denoted  $u, v, w, x, y, z$ , the true ground cover images are denoted by  $a, b, c, d, e, f$  and the landform image is denoted by  $L$ . The circles denote latent (unobserved) images and the squares denote observed images. By analogy with CPNs, the joint distribution of all the images would factorize as

$$p(u|a) p(v|b) p(w|c) p(x|d) p(y|e) p(z|f) \times \\ p(b|a, L) p(c|b, L) p(d|c, L) p(e|d, L) p(f|e, L) p(a|L) p(L) \quad (3.1)$$

Aside from the fact the the distributions are high dimensional e.g.  $u$  and  $a$  have 8000 by 8000 variables, this simply looks like a conditional probability network with latent variables.

To implement such a model we need to have tractable models for the probabilities in (3.1), determine the identifiability of the parameters in the model and have a method for estimating the parameters. Having fitted such a model, we then need to predict the unobserved true images given the observed images. Given the large volumes of data, we also need to have efficient algorithms for doing the calculations.

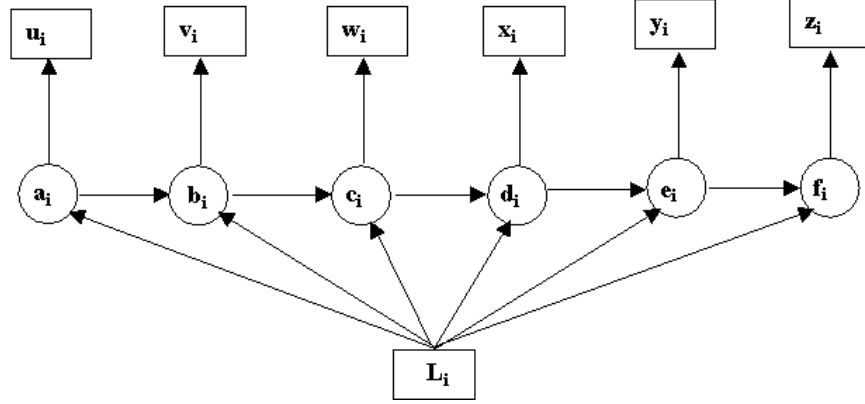


Figure 3.4. CPN model assuming independent pixels

### 3.4.2 Construction of a contextual “CPN” model for raster images

To build a suitable model we put together two pieces, the first piece will consider the relationship between images and the second piece will focus on the within image structure.

The first piece is a CPN for images assuming independent pixels i.e. the same CPN is applied to each pixel in the image data. Let  $u, v, w, x, y, z$  denotes the classified images for years 89, 90, 93, 94, 96, and 98 and  $a, b, c, d, e, f$  denotes the “true” images for the same years. Here, for example

$$u = (u_i, i = 1, \dots, n), a = (a_i, i = 1, \dots, n), z = (z_i, i = 1, \dots, n)$$

and  $n$  is the number of pixels.  $L$  denotes the landform image. The probability density for all the images is

$$\begin{aligned} p &= p(a, b, c, d, e, f, u, v, w, x, y, z, L) \\ &= \prod_{i=1}^n p(u_i|a_i)p(a_i|L_i)p(v_i|b_i)p(b_i|a_i, L_i) \cdots p(z_i|f_i)p(f_i|e_i, L_i)p(L_i) \end{aligned} \quad (3.2)$$

The second piece we will use is a Markov Random Field model for images [2]. First define  $n(i)$  to be the set of neighbours of pixel  $i$  and  $r(i)$  the set of pixels excluding pixel  $i$ , i.e. the rest of the pixels. For concreteness we take the set of neighbours to be the eight nearest neighbours with appropriate modifications at the edges of the image. Writing  $a_{n(i)}$  for the neighbouring set of image values and  $a_{r(i)}$  for the image values at all pixels except pixel  $i$ , these models have the property that the conditional distribution of  $a_i$  given  $a_{r(i)}$  depends only on  $a_{n(i)}$  that is

$$p(a_i|a_{r(i)}) = p(a_i|a_{n(i)})$$



We use similar notation for the other unobserved images. We can make a hybrid model from the two components as follows

$$p^*(a, b, c, d, e, f, u, v, w, x, y, z, L) = \exp \{ \log p + \log p(a) + \cdots + \log p(f) + \log p(u) + \cdots + \log p(z) + \log p(L) \} / U \quad (3.3)$$

where  $p$  is the CPN model (3.2) for the images assuming independent pixels and the remaining terms in brackets correspond to Markov random field models for each of the images. The term  $U$  is simply a normalising constant. Note that (3.3) defines a Gibbs distribution.

To predict the unobserved true maps  $a, b, c, d, e, f$  we want to calculate the conditional probability

$$p^*(a, b, c, d, e, f | u, v, w, x, y, z, L) \quad (3.4)$$

and find the images  $a, b, c, d, e, f$  which maximise this probability. To see how to do this we first do a calculation. Writing

$$(a, b, c, d, e, f) = (a_i, b_i, c_i, d_i, e_i, f_i, a_{r(i)}, b_{r(i)}, c_{r(i)}, d_{r(i)}, e_{r(i)}, f_{r(i)})$$

and using factorisations for each term in (3.3) we get

$$\begin{aligned} p^*(a, b, c, d, e, f | u, v, w, x, y, z, L) &= p^*(a_i, b_i, c_i, d_i, e_i, f_i | a_{r(i)}, b_{r(i)}, c_{r(i)}, d_{r(i)}, e_{r(i)}, f_{r(i)}, u, v, w, x, y, z, L) \\ &\quad \times p^*(a_{r(i)}, b_{r(i)}, c_{r(i)}, d_{r(i)}, e_{r(i)}, f_{r(i)} | u, v, w, x, y, z, L) \\ &= p^*(a_i, b_i, c_i, d_i, e_i, f_i | a_{n(i)}, b_{n(i)}, c_{n(i)}, d_{n(i)}, e_{n(i)}, f_{n(i)}, u, v, w, x, y, z, L) \\ &\quad \times p^*(a_{r(i)}, b_{r(i)}, c_{r(i)}, d_{r(i)}, e_{r(i)}, f_{r(i)} | u, v, w, x, y, z, L) \end{aligned}$$

Given this result, a cyclic ascent algorithm for doing the maximisation can be constructed as follows:

1. Start with initial estimates of the unobserved images  $a, b, c, d, e, f$  e.g. by using the results for the independent pixels case
2. Visit each pixel  $i$  in turn keeping all labels fixed except  $a_i, b_i, c_i, d_i, e_i, f_i$  and compute

$$\begin{aligned} p^*(a, b, c, d, e, f | u, v, w, x, y, z, L) &= p^*(a_i, b_i, c_i, d_i, e_i, f_i | a_{n(i)}, b_{n(i)}, c_{n(i)}, d_{n(i)}, e_{n(i)}, f_{n(i)}, u, v, w, x, y, z, L) \\ &\quad \times p^*(a_{r(i)}, b_{r(i)}, c_{r(i)}, d_{r(i)}, e_{r(i)}, f_{r(i)} | u, v, w, x, y, z, L) \end{aligned}$$

It can be shown [13] that

$$p^*(a_i, b_i, c_i, d_i, e_i, f_i | a_{n(i)}, b_{n(i)}, c_{n(i)}, d_{n(i)}, e_{n(i)}, f_{n(i)}, u, v, w, x, y, z, L)$$

factorises so that it can be computed from a CPN with a graph augmented with dummy neighbourhood nodes as in Figure 3.5. Choose

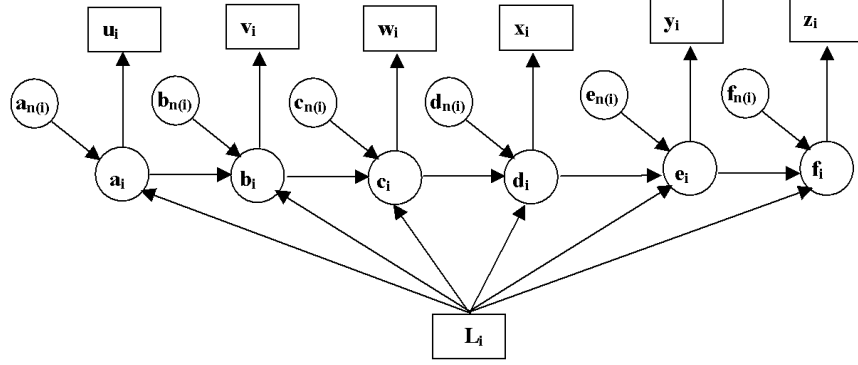


Figure 3.5. CPN augmented with neighbour information

map classes  $a_i, b_i, c_i, d_i, e_i$  and  $f_i$  to maximise (3.1). Another strategy at this point is to choose labels for the unobserved maps individually by maximising the marginal distributions

$$p^*(a_i | a_{n(i)}, b_{n(i)}, c_{n(i)}, d_{n(i)}, e_{n(i)}, f_{n(i)}, u, v, w, x, y, z, L)$$

and similarly for  $b_i, c_i, d_i, e_i$ , and  $f_i$ .

3. Continue cycling over all pixels until convergence. For the present application we only do a few iterations.

### 3.4.3 Estimation of parameters

To use the algorithm in practice requires the specification of parameters in the model. The specific Markov random field model we used for each unobserved image was

$$p(a_i | a_{n(i)}) = \exp\{\alpha + \beta N(a_i)\} \quad (3.5)$$

where  $N(a_i)$  is the number of 8 nearest neighbours of pixel  $i$  with label  $a_i$ , see for example [2]. For parameter values we used  $\alpha = 0$  and  $\beta = 1$ . These parameters can be varied to change the relative weighting between image data and contextual information. The probabilities  $p(u_i | a_i), \dots, p(z_i | f_i)$  are assumed to be the same for each pixel (within a zone) and are estimated from error rates derived from the classification process. The remaining probabilities in (3.1) are estimated by the EM algorithm [9],[4] ignoring spatial dependence. Ignoring this dependence should not be too critical since we have large sample sizes. The transition probabilities for the ground cover classes are identifiable, see the Appendix. Alternative estimation procedures such as coding or pseudo likelihood could also be used [1, 2].

The EM algorithm is implemented as follows:

1. Guess initial values for all the unknown probabilities in (3.2), e.g., by random generation.
2. Perform the E step. For the model of Figure 3.4 this could be done by calculating the table of expected counts

$$m(a_i, b_i, c_i, d_i, e_i, f_i | u_i, v_i, w_i, x_i, y_i, z_i, L_i) = \sum_{i=1}^n p(a_i, b_i, c_i, d_i, e_i, f_i | u_i, v_i, w_i, x_i, y_i, z_i, L_i) \quad (3.6)$$

where the conditional probability is computed using the model (3.2). Next, we could then calculate the expected marginal tables of counts defined by  $m(a_i, L_i)$ ,  $m(b_i, a_i, L_i)$ ,  $m(c_i, b_i, L_i)$ ,  $m(d_i, c_i, L_i)$ ,  $m(e_i, d_i, L_i)$ ,  $m(f_i, e_i, L_i)$ , where for example arguments not appearing in  $m(\cdot)$  are summed over in (3.6). However this can be done more efficiently, see [17].

3. Perform the M step. Estimate probabilities by calculating relative frequencies assuming the expected counts were observed data. For example  $p(a_i | L_i) = m(a_i, L_i) / m(L_i)$  and  $p(b_i | a_i, L_i) = m(b_i, a_i, L_i) / m(a_i, L_i)$ .
4. Go to 2 until convergence.

#### 3.4.4 A model for handling uncertainty in input class labels

When computing the probability of true class labels from the model in Figure 3.4, it is assumed that class labels are known. However, the Gaussian maximum likelihood classifier also produces posterior probabilities of class membership and we can use these in the mapping process. A graphical representation of a model for doing this is given in Figure 3.6.

In Figure 3.6 variable  $u_i^*$ , for example, refers to the six bands of the satellite image at pixel  $i$  on the first date in the series. Spatial context could also be included in the same manner as in Figure 3.5. Details about the modifications to the usual calculations are given in the Appendix.

### 3.5 An example

An example of a classification for a single date produced by using the methods described above is given in Figure 3.7. The figure is represented in gray scale, however colour would be more impactful.

To provide a (salinity) accuracy assessment for the classification, 124 validation sites were visited, and their salinity status noted. The results are given in Table 3.1 below. Of the 124 sites 6 were incorrectly labeled and 118 were correctly labeled, or in other words an overall salinity mapping

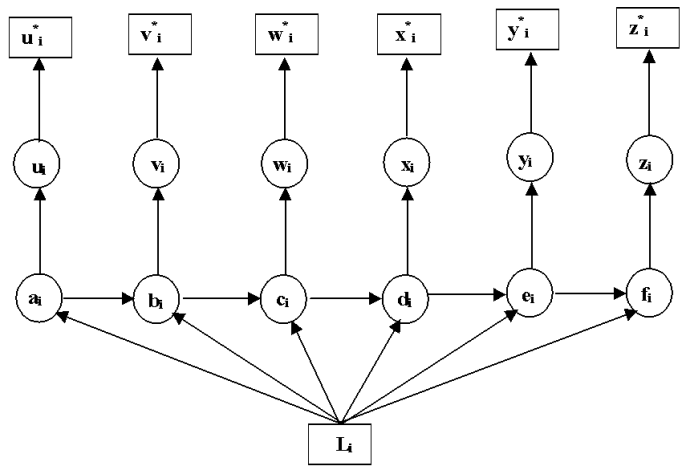


Figure 3.6. Modified CPN for handling uncertainty in input classification maps

Table 3.1. Accuracy assessment for example region

		Truth	
Map		Saline	non-saline
	Saline	40	5
	Non-saline	1	78

accuracy of approximately 95%. For the broader region, estimates of the percentage of this catchment affected by salinity range from 0 to 9.8%. Maps such as in Figure 3.7 are available for six dates for this area.

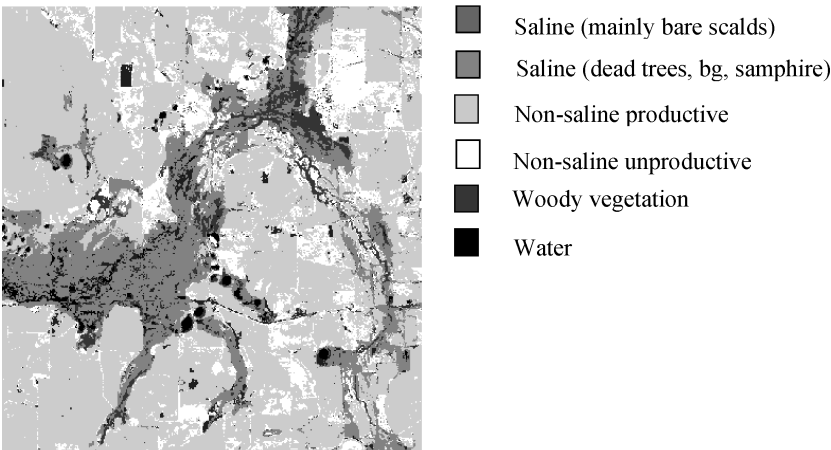


Figure 3.7. Example single date “true” map

There are of course significant issues in how to assess the accuracy of such large scale maps given limited resources, however we will not go into these here.

In conclusion, the methods discussed in this chapter seem promising for handling large scale environmental monitoring problems involving the use of time series of satellite imagery. A particularly desirable feature of the methods is the ability to propagate uncertainties in the inputs to the output products.

### 3.6 Appendix: Modifications to calculations when there is uncertainty in the classification input maps

We consider the independent pixels case and omit the subscript  $i$  on the variables. For generality and compactness we represent the classified satellite image class at pixel  $i$  at times  $1, 2, 3, \dots$  as  $u_1, u_2, u_3, \dots$  and the true classes as  $a_1, a_2, a_3, \dots$ .

The joint distribution is

$$p(u_1, \dots, u_p, a_1, \dots, a_p, L) = \prod_{j=1}^p p(u_j^* | u_j) p(u_j | a_j) p(a_1, \dots, a_p, L)$$

From this we can see that

$$\begin{aligned} p(a_1, \dots, a_p | u_1^*, \dots, u_p^*, L) \\ \propto \prod_{j=1}^p \left[ \sum_{u_j} p(u_j^* | u_j) p(u_j | a_j) \right] p(a_1, \dots, a_p, L) \end{aligned} \quad (3.7)$$

Writing

$$p(u_j | u_j^*) = \frac{p(u_j^* | u_j)}{\sum_{u_j} p(u_j^* | u_j)}$$

for the posterior probability obtained from the Gaussian maximum likelihood classifier using uniform prior probabilities (3.7) can be written as

$$\begin{aligned} p(a_1, \dots, a_p | u_1^*, \dots, u_p^*, L) \\ \propto \prod_{j=1}^p \left[ \sum_{u_j} p(u_j | u_j^*) p(u_j | a_j) \right] p(a_1, \dots, a_p, L) \end{aligned} \quad (3.8)$$

When  $p(u_j | u_j^*)$  is an indicator vector i.e. zero everywhere except in one position (3.8) reduces to the usual formula for  $p(a_1, \dots, a_p | u_1, \dots, u_p, L)$ .

Note that all calculations can be done efficiently using the algorithm in [15]. See also [16]. We can also include spatial dependence as in Figure 3.5.

### 3.7 Appendix: Identifiability of the CPN model in the independent pixels case

**Theorem 2.** *If the  $G$  by  $G$  matrices  $A_i$  with elements defined by  $p(u_i|a_i)$  for  $u_i, a_i = 1, \dots, G$  are known and full rank for all  $i$  then  $p(a_1, \dots, a_p, L)$  can be determined from the distribution  $q(u_1, \dots, u_p, L)$  of the observed variables.*

*Proof.* We use induction on  $p$ . First when  $p = 1$  we have

$$\sum_{a_1} p(u_1|a_1)p(a_1, l) = q(u_1, l)$$

where  $u_1 = 1, \dots, G$  and  $l = 1, \dots, L$ , say. This is a linear equation which can be solved uniquely for  $p(a_1, l)$  since the matrix with elements  $p(u_1|a_1)$  is full rank.

Next assume the result is true for  $p$  and we will demonstrate that it is true for  $p + 1$ . We have

$$\sum_{a_1, \dots, a_{p+1}} \prod_{i=1}^{p+1} p(u_i|a_i)p(a_1, \dots, a_{p+1}, l) = q(u_1, \dots, u_{p+1}, l) \quad (3.9)$$

Summing both sides of this equation over  $u_{p+1}$  gives

$$\sum_{a_1, \dots, a_p} \prod_{i=1}^p p(u_i|a_i)p(a_1, \dots, a_p, l) = q(u_1, \dots, u_p, l) \quad (3.10)$$

where we omit arguments of  $q$  which have been summed over. By the inductive hypothesis it follows that we can obtain  $p(a_1, \dots, a_p, l)$ . It remains to show that we can obtain  $p(a_p, a_{p+1}, l)$ . This is sufficient since the joint distribution is defined by its marginals over the cliques, see [8] and Figure 3.4. Summing (3.9) over  $u_1, \dots, u_{p-1}$  gives

$$\sum_{a_p, a_{p+1}} p(u_{p+1}|a_{p+1})p(u_p|a_p)p(a_p, a_{p+1}, l) = q(u_p, u_{p+1}, l) \quad (3.11)$$

Now writing  $B(l)$  for the matrix with elements  $p(a_p, a_{p+1}, l)$  and  $Q(l)$  for the matrix  $q(u_p, u_{p+1}, l)$  (3.11) can be written as the matrix equation

$$A_p B(l) A_{p+1} = Q(l)$$

Where  $A_p$  and  $A_{p+1}$  are square and full rank. Hence the result follows.  $\square$

## References

- [1] Besag, J. E. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society B*, 36, 1974, pp. 192-326.
- [2] Besag, J. E. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B* 48, 1986, pp. 259-302.
- [3] Caccetta, P., Campbell, N., West, G., Kiiveri, H., and Gahegan, M. Aspects of reasoning with uncertainty in an agricultural GIS environment. *The New Review of Applied Expert Systems* 1, 1995, pp. 161-177.
- [4] Caccetta, P. Remote Sensing, GIS and Bayesian Knowledge-based Methods for Monitoring Land Condition. PhD thesis, Department of Computer Science, Curtin University of Technology, Western Australia, 1997.
- [5] Caccetta, P. C., Campbell, N. A. C., Evans, F., Furby, S. L., Kiiveri, H. T., and Wallace, J. F. (2000). Mapping and monitoring land use and condition change in the south west of Western Australia using remote sensing and other data. In *Proceedings of the Europa 2000 Conference*, Barcelona.
- [6] Campbell, N. A. and Atchley, W. R. (1981), 'The geometry of canonical variate analysis', *Syst. Zoology*, Vol. 30, No. 3, pp. 268-280.
- [7] Subpixel matching using cross correlation and second derivatives. Submitted to *ISPRS Journal of Photogrammetry and Remote Sensing*.
- [8] Darroch, J. N., Lauritzen, S. L. and Speed, T. P. Log-linear models for contingency tables and Markov fields over graphs. *Annals of Statistics* 8, 1980, pp. 522-539.
- [9] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* 39, 1977, pp.1-21.
- [10] Furby, S. L., (1994) Discriminating between pasture and barley grass and saltbush using multi-temporal imagery. CMIS technical report.
- [11] Furby, S. L. and Campbell (2001), 'Calibrating images from different dates to like value digital counts', *Remote Sensing of the Environment*, 77, 186-196.
- [12] Jensen, F. V. *An Introduction to Bayesian Networks*. Springer Verlag, New York, 1996.

- [13] Kiiveri, H. T. and Caccetta, P. Data fusion, uncertainty and causal probabilistic networks for monitoring the salinisation of farmland. *Digital signal processing*, 8, 225-230
- [14] Kiiveri, H. T. Some statistical models for remotely sensed data. In *SISC96 Imaging Interface Workshop Proceedings*, 1996.
- [15] Lauritzen, S. L., and Spiegelhalter, D. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B* 50, 1988, pp. 157-224
- [16] Lauritzen, S. L. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association* 87, 1992, pp. 1098-1108
- [17] Lauritzen, S. L. (1995). 'The EM algorithm for graphical association models with missing data', *Computational Statistics and Data Analysis*, 19, pp. 191-201.
- [18] NASA, (2001). Landsat 7 Science data users handbook. Available on line at [http://ltpwww.gsfc.nasa.gov/IAS/handbook/handbook\\_toc.html](http://ltpwww.gsfc.nasa.gov/IAS/handbook/handbook_toc.html).
- [19] Rao, C. R. (1966), *Linear statistical inference and its applications*. Second Edition, Wiley, New York.



<http://www.springer.com/978-0-387-95471-4>

Nonlinear Estimation and Classification

Denison, D.D.; Hansen, M.H.; Holmes, C.C.; Mallick, B.;  
Yu, B. (Eds.)

2003, VII, 477 p., Softcover

ISBN: 978-0-387-95471-4