Learning Compatibility Coefficients for Relaxation Labeling Processes

Marcello Pelillo, *Member, IEEE,* and Mario Refice, *Member, IEEE*

Abstract—**Relaxation labeling processes have been widely used in many different domains including image processing, pattern recognition, and artificial intelligence. They are iterative procedures that aim at reducing local ambiguities and achieving global consistency through a parallel exploitation of contextual information, which is quantitatively expressed in terms of a set of "compatibility coefficients." The problem of determining compatibility coefficients has received a considerable attention in the past and many heuristic, statistical-based methods have been suggested. In this paper, we propose a rather different viewpoint to solve this problem: we derive them attempting to optimize the performance of the relaxation algorithm over a sample of training data; no statistical interpretation is given: compatibility coefficients are simply interpreted as real numbers, for which performance is optimal. Experimental results over a novel application of relaxation are given, which prove the effectiveness of the proposed approach.**

Index Terms— **Compatibility coefficients, constraint satisfaction, gradient projection, learning, neural networks, nonlinear programming, part-of-speech disambiguation, relaxation labeling.**

I. Introduction

MANY problems in pattern recognition and artificial intelligence involve labeling a set of objects in such a way that existing domain-specific constraints are satisfied. They are commonly referred to as consistent labeling (or constraint satisfaction) problems [1], and have been proved to be NP-complete [2]. Basically, there are two ways of representing the constraint (or world) model. The first consists simply of specifying the set of allowable object-label configurations. This is the discrete problem, which can be solved, for example, by an iterative label discarding algorithm [3]. In a more general approach, instead, labels are no longer either completely compatible or incompatible; rather, the constraints are expressed in terms of real-valued *compatibility coefficients* that measure the strength of agreement for N-tuples of object-label pairs (in practice $N = 2$, but some experiments with higher order compatibilities have been attempted [4]–[6]). This is referred to as the continuous labeling problem, and can be solved by means of a (continuous) relaxation labeling process [3], [7]. This paper is primarily concerned with the continuous problem, but the approach advocated here can be adopted for the discrete problem as well.

Relaxation labeling processes are iterative procedures that aim at reducing labeling ambiguities and achieving global consistency through a parallel exploitation of local information. They are given as input an initial ambiguous labeling assignment, and iteratively update it taking into account the compatibility model. The iterative nature of relaxation processes allows information to propagate, until global consistency is achieved. The notion of consistency for the continuous model has been precisely defined by Hummel and Zucker [7], but alternative definitions can be formulated [8].

The attractive feature of relaxation processes, which strongly relates them to certain connectionist models [9], [10] is that a global, complex task is accomplished by means of simple, local computations. This is what Fahlman *et al.* [11] call *massively parallel architectures.* Actually, the majority of the applications of relaxation labeling have been implemented on serial machines, but recently parallel architectures have been developed [12], [13], for attempting to exploit the parallel nature of relaxation.

It is widely recognized that the behavior of relaxation processes is greatly affected by the choice of compatibility coefficients. Haralick *et al.* [14], for example, derived analytical expressions for the fixed points of relaxation, in terms of the compatibilities, and introduced the notion of "no-information" fixed point for attempting to study possible biases of coefficients. In [15], instead, O'Leary and Peleg studied a simple but significant case in which only two objects and two labels are involved; they analyzed three different relaxation schemes and showed how certain choices of compatibility coefficients result in an unacceptable behavior of the algorithms, as they yield a result independent of the initial labeling. They conjectured that the same situation occurs also in the general case, where more objects and more labels are given, and emphasized the importance of having guidelines for determining compatibilities.

A number of studies have been conducted over the last years, aimed at providing suitable interpretations for compatibilities and devising methods to derive them. In their seminal paper, for example, Rosenfeld *et al.* [3] proposed the use of correlation coefficient, on the basis of simple empirical considerations; later, Peleg and Rosenfeld [16] suggested also mutual information, and developed a method of deriving the coefficients directly from the initial ambiguous labeling assignment; this "unsupervised" method has been widely applied especially within the machine vision domain.

Manuscript received September 28, 1992; revised September 20, 1993. Recommended for acceptance by Associate Editor S. Peleg.

M. Pelillo is with the Dipartimento di Informatica, Università di Bari, Via G. Amendola 173, 70126 Bari, Italy.

M. Refice is with the Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via E. Orabona 4, 70125 Bari, Italy.

IEEE Log Number 9401640.

Yamamoto [17] proposed another form for compatibilities based on heuristic modification of mutual information. In [18], Peleg derived a theoretical-based relaxation scheme in which compatibility coefficients were found to be a normalized conditional probability measure; this measure was also suggested by Davis and Rosenfeld [19]. More recently, Kittler [20], within a similar statistical framework, developed a general "evidence combining formula" and argued that the choice of compatibility coefficients depends only on the interaction relation over objects and labels of the problem at hand. He showed that under appropriate assumptions about interaction relations, well-known relaxation schemes can be derived from his formulas. A different, nonstatistical approach was proposed by Hummel [21] who suggested to determine compatibility coefficients so that given sample labelings become consistent, according to the definition of consistency defined in [7]. His approach, however, requires solving a large system of linear inequalities, that can be even incompatible. Also, the presence of undesirable "spurious" labelings should be avoided; he discussed how to overcome this difficulty over simple toy examples. Finally, it is worth mentioning that in some practical applications compatibilities have been subject to ad-hoc, problem-dependent choices (see e.g., [22], [23]).

In contrast with the standard statistical approach, this paper presents a somewhat different viewpoint for the compatibility coefficients: we propose to derive them attempting to optimize the performance of the relaxation algorithm over a sample of training data. No statistical interpretation is given: compatibility coefficients are simply interpreted as real numbers, for which relaxation labeling attains optimal performance. More specifically, we formulate the problem of determining the coefficients as a nonlinear programming problem that can be solved by standard gradient-descent methods. In particular, we develop an iterative algorithm based on the gradient projection method of Rosen [24], [25]. The resulting process can be regarded as a *learning* process since the performance of relaxation improves over time. Optimization methods have been widely used in connection with relaxation labeling [7], [8], [26], [27] especially for attempting to provide theoretical justifications to the algorithm. However, the novelty of our approach relies on the use of optimization techniques not for the development of a new relaxation scheme, but for the determination of an optimal set of compatibility coefficients, once that a relaxation scheme has been previously chosen.

The idea of determining the parameters of a model from data by minimizing (or maximizing) an appropriate objective function is of course an old one, and it is the basis of many statistical methods such as regression analysis and maximum-likelihood estimation. More recently, the same optimization approach has proved to be successful in estimating the parameters of a Markov chain [28], and it has become the standard approach to train artificial neural networks [29]. The latter is especially interesting owing to the close relationship between the fields of neural networks and relaxation labeling.

The paper is organized as follows. We begin by briefly describing relaxation labeling processes in Section II. Section III, instead, formulates the problem of learning compatibility coefficients as a nonlinear programming problem and Section IV develops an algorithm for solving the problem based on the gradient projection method, giving recursive formulas for evaluating the gradient of the objective function. In Section V, a comparison with neural network learning theory is outlined, and Section VI presents some results over a practical application of relaxation labeling. Finally, Section VII concludes the paper.

II. Relaxation Labeling Processes

Relaxation labeling literature contains a variety of different algorithms (a survey and an extensive bibliography is given in [30]); in this paper, however, we will refer to the standard Rosenfeld *et al.*'s formulas [3], although the same arguments can well be applied to other schemes.

Consider a set of objects $\boldsymbol{B} = \{b_1, \cdots, b_n\}$, and a set of labels $\Lambda = \{1, \cdots, m\}$. It is our purpose to label each object of $\boldsymbol{B}$ with exactly one label of Λ. By means of some local measurement we can derive, for each object b_i, a vector $\boldsymbol{p}_i^{(0)} = (p_{i1}^{(0)}, \cdots, p_{im}^{(0)})^T$ such that $0 \leq p_{i\lambda}^{(0)} \leq 1$, for $i = 1 \cdots n$, and $\lambda = 1 \cdots m$, and $\Sigma_\lambda p_{i\lambda}^{(0)} = 1$, for $i = 1 \cdots n$. Each $\boldsymbol{p}_i^{(0)}$ can be interpreted as the *a priori* probability distribution of labels for the object b_i. By simply concatenating $\boldsymbol{p}_1^{(0)}, \boldsymbol{p}_2^{(0)}, \cdots, \boldsymbol{p}_n^{(0)}$ we obtain an initial weighted labeling assignment for the objects of $\boldsymbol{B}$ that will be denoted by $\boldsymbol{p}^{(0)} \in R^{nm}$. Also, it is supposed that labels do not occur independently on each other, but are subject to contextual constraints, which are expressed in terms of an $n \times n$ block matrix R

$$R = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{n1} & \cdots & R_{nn} \end{bmatrix},$$

where each R_{ij} is an $m \times m$ matrix of nonnegative real-valued compatibility coefficients:

$$R_{ij} = \begin{bmatrix} r_{ij}(1,1) & \cdots & r_{ij}(1,m) \\ \vdots & \ddots & \vdots \\ r_{ij}(m,1) & \cdots & r_{ij}(m,m) \end{bmatrix}.$$

The coefficient $r_{ij}(\lambda, \mu)$ measures the strength of compatibility between λ on object b_i and μ on object b_j: high values correspond to compatibility and low values to incompatibility.

The relaxation algorithm accepts as input the initial labeling $\boldsymbol{p}^{(0)} = (\boldsymbol{p}_1^{(0)T}, \cdots, \boldsymbol{p}_n^{(0)T})^T$, and updates it iteratively according to the constraint model in order to achieve global consistency.[1] At the tth step the labeling is updated according to the following formula:

$$p_{i\lambda}^{(t+1)} = \frac{p_{i\lambda}^{(t)} q_{i\lambda}^{(t)}}{\sum_{\mu=1}^{m} p_{i\mu}^{(t)} q_{i\mu}^{(t)}}, \tag{1}$$

[1] We mention that the original definition of consistency of Rosenfeld *et al.* [3] is somewhat different from that developed later by Hummel and Zucker [7]. However, recently, a correspondence between these two notions have been established [31].

where the denominator is simply a normalization factor, and

$$q_{i\lambda}^{(t)} = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_{j\mu}^{(t)} \tag{2}$$

represents the strength of support that context gives to λ for being the correct label for b_i, at step t.

Ideally, the relaxation process should evolve until a fixed point is reached; in practice, however, the algorithm is generally stopped when some termination criterion is satisfied. For example, it can terminate when the distance between successive labelings becomes small enough or, more commonly, after a fixed number of steps. The final labeling can be used to label the objects of $\boldsymbol{B}$ according to a maxima selection criterion [32].

In practical applications, some simplifying assumptions are made. First, it is usually assumed that objects interact only within a small neighborhood; for each $i = 1 \cdots n$ we will denote by Δ_i the neighborhood of object b_i, that is the set of relative positions that are supposed to influence the object on site i. Note that our definition of neighborhood is slightly different from the usual one. In our notation, Δ_i is a set of "relative positions" (or offsets); thus, for example, if the neighborhood of object b_i consists of its immediate predecessor and successor, we have $\Delta_i = \{-1, +1\}$. Second, it is generally supposed that compatibility coefficients are "stationary," in the sense that do not depend on the absolute position of objects but, rather, on their relative distance. This is formally expressed with the relation $R_{ij} = R_{hk}$, for $j - i = k - h$. Having made these simplifying assumptions, only $|\Delta|$ compatibility matrices are needed, where $\Delta = \cup_{i=1}^{n} \Delta_i$ and $|\cdot|$ denotes the cardinality of a set, provided that the support formula (2) is replaced with

$$q_{i\lambda}^{(t)} = \sum_{\delta \in \Delta_i} \sum_{\mu=1}^{m} r_{\delta\lambda\mu} p_{i+\delta,\mu}^{(t)}. \tag{3}$$

Now, $r_{\delta\lambda\mu}$ denotes the compatibility between labels λ and μ, when μ is at offset δ from λ.

We will find it convenient to abandon the matrix notation for compatibilities, regarding them as real vectors: $\boldsymbol{r} \in R^D$, with $D = |\Delta| m^2$.

III. Formulation of the Problem

The learning algorithm developed in this paper is based on the assumption that a set of instances of the problem we intend to solve is available. To be more specific, it is supposed that a number of learning samples exist:

$$L = \{L_1, \cdots, L_N\},$$

where each sample $L_\gamma (\gamma = 1 \cdots N)$ is a set of labeled objects of the form

$$L_\gamma = \{(b_i^\gamma, \lambda_i^\gamma) : 1 \le i \le n_\gamma, b_i^\gamma \in \boldsymbol{B}, \lambda_i^\gamma \in \Lambda\}.$$

Clearly, the b_i's can well be feature vectors describing real objects.

For each $\gamma = 1 \cdots N$ let $\boldsymbol{p}^{(L_\gamma)} \in R^{n_\gamma m}$ denote the unambiguous labeling assignment for the objects of L_γ, that is

$$p_{i\alpha}^{(L_\gamma)} = \begin{cases} 0, & \text{if } \alpha \neq \lambda_i^\gamma, \\ 1, & \text{if } \alpha = \lambda_i^\gamma. \end{cases}$$

Furthermore, suppose that we have some mechanism for constructing an initial labeling $\boldsymbol{p}^{(I_\gamma)}$ on the basis of the objects in L_γ, and let $\boldsymbol{p}^{(F_\gamma)}$ denote the labeling produced by the relaxation algorithm when $\boldsymbol{p}^{(I_\gamma)}$ is given as input. The same mechanism for deriving the initial labelings should be used both in the "learning" and in the "testing" phases. In order to clarify our discussion, we need to specify that, for the proposed algorithm to work, the final labeling $\boldsymbol{p}^{(F_\gamma)}$ is by no means required to be the converged, fixed-point solution of the relaxation process; rather, it is simply the labeling obtained employing an appropriate stopping criterion whatsoever. In the experiments reported in this paper, for example, the criterion was to stop relaxation after a fixed number of steps.

A relaxation process is a function that, given as input a vector of compatibility coefficients $\boldsymbol{r}$ and an initial labeling $\boldsymbol{p}^{(I)}$, produces iteratively the final labeling $\boldsymbol{p}^{(F)}$, i.e., $\boldsymbol{p}^{(F)} \leftarrow \text{Relax}(\boldsymbol{r}, \boldsymbol{p}^{(I)})$. In our approach we consider the relaxation operator as a function of the compatibility coefficients only, the initial labeling being considered as a constant. To emphasize this dependence we will write $p_{i\lambda}^{(F)}(\boldsymbol{r})$ to denote the $i\lambda$ component of the final labeling.

Within this framework, a natural way to derive compatibility coefficients is to choose them so that $\boldsymbol{p}^{(F_\gamma)}$ be as close as possible to the desired labeling $\boldsymbol{p}^{(L_\gamma)}$, for each $\gamma = 1 \cdots N$. To do so, we can define an error function measuring the loss incurred when $\boldsymbol{p}^{(F_\gamma)}$ is obtained instead of $\boldsymbol{p}^{(L_\gamma)}$, and attempt to minimize it. As an example, a quadratic error function may be adopted:

$$E_\gamma^{(Q)}(\boldsymbol{r}) = \frac{1}{2} \sum_{i=1}^{n_\gamma} \sum_{\lambda=1}^{m} (p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r}) - p_{i\lambda}^{(L_\gamma)})^2, \tag{4}$$

which measures the (squared) Euclidean distance between $\boldsymbol{p}^{(L_\gamma)}$ and $\boldsymbol{p}^{(F_\gamma)}$, when $\boldsymbol{r}$ is used. Also, the total error achieved can be defined as

$$E^{(Q)}(\boldsymbol{r}) = \sum_{\gamma=1}^{N} E_\gamma^{(Q)}(\boldsymbol{r}). \tag{5}$$

An alternative error function comes from information theory. Notice, in fact, that both $\boldsymbol{p}^{(F_\gamma)}$ and $\boldsymbol{p}^{(L_\gamma)}$ are composed of n_γ discrete probability distributions: $\boldsymbol{p}_i^{(F_\gamma)}$ and $\boldsymbol{p}_i^{(L_\gamma)}$, respectively ($i = 1 \cdots n_\gamma$). Of course, we wish that each $\boldsymbol{p}_i^{(F_\gamma)}$ be as close as possible to $\boldsymbol{p}_i^{(L_\gamma)}$. A well-known information–theoretic divergence measure between two probability distributions is Kullback's I directed divergence [33], which has been successfully employed also in certain connectionist learning procedures [34], [35]. Kullback's divergence is defined as[2]

$$I(\boldsymbol{p}_i^{(L_\gamma)} | \boldsymbol{p}_i^{(F_\gamma)}(\boldsymbol{r})) = \sum_{\lambda=1}^{m} p_{i\lambda}^{(L_\gamma)} \ln \frac{p_{i\lambda}^{(L_\gamma)}}{p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r})}. \tag{6}$$

[2]We have used natural logarithms but, of course, any other base can be used.

Since $\boldsymbol{p}_i^{(L_\gamma)}$ is a simple class indicator vector, containing all zeros except at the position corresponding to λ_i^γ, (6) reduces to

$$I(\boldsymbol{p}_i^{(L_\gamma)} | \boldsymbol{p}_i^{(F_\gamma)}(\boldsymbol{r})) = -\ln p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r}). \tag{7}$$

It turns out that $I \geq 0$, with equality, if and only if the two distributions are identical [33]. In addition, I is not symmetric. Nevertheless, as pointed out in [34], this is a reasonable property, as the difference between the actual and the obtained distributions is weighted by the actual probabilities $p_{i\lambda}^{(L_\gamma)}$'s.

In order for I to be well defined, it is required that $\boldsymbol{p}_i^{(L_\gamma)}$ be *absolutely continuous* with respect to $\boldsymbol{p}_i^{(F_\gamma)}$ [33]. This means that $p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r}) \neq 0$ whenever $p_{i\lambda}^{(L_\gamma)} \neq 0$ or, in our case,

$$p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r}) \neq 0, \qquad \text{for } i = 1 \cdots n_\gamma;$$

this amounts to requiring that the relaxation algorithm assigns nonzero probability to the correct labels. In practice, this situation should occur rarely as long as the initial labelings $\boldsymbol{p}^{(I_\gamma)}$'s are accurate enough, but even in this case one can use a recent divergence measure proposed by Lin [36] that does not suffer from this drawback and is closely related to Kullback's measure.

The "logarithmic" error achieved for sample γ is

$$E_\gamma^{(I)}(\boldsymbol{r}) = -\sum_{i=1}^{n_\gamma} \ln p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r}) \tag{8}$$

and the total error is

$$E^{(I)}(\boldsymbol{r}) = \sum_{\gamma=1}^{N} E_\gamma^{(I)}(\boldsymbol{r}). \tag{9}$$

In the following, E will be used to denote either $E^{(Q)}$ or $E^{(I)}$.

Now, the problem of determining the compatibility coefficients can be restated as the problem of minimizing the function E with respect to $\boldsymbol{r}$. More formally, the problem we wish to solve is

$$\begin{aligned} &\text{minimize } E(\boldsymbol{r}), \boldsymbol{r} \in R^D \\ &\text{subject to } r_{d\alpha\beta} \geq 0. \end{aligned} \tag{10}$$

This is a nonlinear programming problem with linear constraints that can be solved by any gradient-descent method [25].

IV. The Learning Algorithm

One popular algorithm for solving linearly constrained minimization problems is Rosen's *gradient projection method* [24]. It is basically an extension of the steepest descent procedure for unconstrained problems, to accommodate the presence of constraints. Here, we make use of a simplified form of the algorithm developed in [25].

The algorithm begins with an initial feasible point $\boldsymbol{r}^{(0)}$ and iteratively produces a sequence of points $\{\boldsymbol{r}^{(k)}\}$ so that the objective function E decreases:

$$E(\boldsymbol{r}^{(k+1)}) \leq E(\boldsymbol{r}^{(k)}). \tag{11}$$

At the kth stage a new point is derived according to the following formula

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \rho_k \boldsymbol{u}^{(k)}, \tag{12}$$

where $\boldsymbol{u}^{(k)}$ is the projection of the gradient of the objective function E onto the intersection of hyperplanes defined by the active constraints (i.e. the constraints that are satisfied as equalities by the current point $\boldsymbol{r}^{(k)}$), and ρ_k is a suitable positive step length, determined so that the new point remains feasible.

Suppose that q constraints are active at step k; the projection of the gradient is accomplished by means of a projection matrix

$$M_k = I - A_k^T (A_k A_k^T)^{-1} A_k, \tag{13}$$

where A_k is a $q \times D$ matrix, the rows of which are the coefficients corresponding to the active constraints, and I is the $D \times D$ identity matrix. Therefore, we have

$$\boldsymbol{u}^{(k)} = M_k \nabla E(\boldsymbol{r}^{(k)}). \tag{14}$$

If $\boldsymbol{u}^{(k)} \neq \boldsymbol{0}$ a suitable step length is determined and the next point is derived according to (12). If, instead, $\boldsymbol{u}^{(k)} = \boldsymbol{0}$ the q-dimensional vector

$$\boldsymbol{\eta}^{(k)} = (A_k A_k^T)^{-1} A_k \nabla E(\boldsymbol{r}^{(k)}) \tag{15}$$

is constructed. If $\eta_j^{(k)} \geq 0$ for all j corresponding to active inequalities,[3] the algorithm stops as the Kuhn–Tucker conditions are satisfied [25]. Otherwise, the inequality constraint corresponding to the most negative component is dropped from the set of active constraints, and the algorithm continues. A thorough discussion of the algorithm can be found in [25].

In general, computing the projection matrix M_k is not a trivial task as a matrix inversion is to be performed. Rosen, in his original paper [24], developed some recursive formulas that greatly reduced the computations involved. Fortunately, owing to the very simple form of constraints, the calculation of the projection matrix as well as of the vector $\boldsymbol{\eta}^{(k)}$ for our problem is straightforward. In fact, suppose that, at step k, q constraints are active, and denote by $i_1, \cdots, i_q$ the corresponding indices.[4] The matrix A_k has the form $[\boldsymbol{e}_{i_1} \boldsymbol{e}_{i_2} \cdots \boldsymbol{e}_{i_q}]^T$ (here the $\boldsymbol{e}_j$'s are the standard basis vectors of R^D), so that $A_k A_k^T = I = (A_k A_k^T)^{-1}$, and the projection matrix $M_k = (m_{ij}^{(k)})$ is as follows:

$$m_{ij}^{(k)} = \begin{cases} 1, & \text{if } i = j \wedge i \text{ is not the index of} \\ & \qquad \text{some active constraint;} \\ 0, & \text{otherwise.} \end{cases}$$

As for the vector $\boldsymbol{\eta}^{(k)}$, it is simple to verify that it contains the components of the gradient corresponding to the active constraints:

$$\eta_j^{(k)} = \boldsymbol{e}_{i_j}^T \nabla E(\boldsymbol{r}^{(k)}), \qquad \text{for } j = 1 \cdots q. \tag{16}$$

In summary, the proposed algorithm for learning compatibilities is as follows.

[3]Notice that problem (10) involves inequality constraints only.

[4]For notational simplicity we use simple subscripts, rather than multi-indexed subscripts.

Algorithm 1:

```
Input: An initial feasible compatibility vector $\boldsymbol{r}^{(0)}$;
Output: An "optimal" compatibility vector.
1) $k := 0$;
2) determine the indices of active constraints, that is $J^{(k)} = \{(d,\alpha,\beta) : r^{(k)}_{d\alpha\beta} = 0\}$;
3) evaluate the vector $\boldsymbol{u}^{(k)}$, as follows:
   $u^{(k)}_{d\alpha\beta} = \begin{cases} \frac{\partial E(\boldsymbol{r}^{(k)})}{\partial r_{d\alpha\beta}}, & \text{if } (d,\alpha,\beta) \notin J^{(k)}, \\ 0, & \text{if } (d,\alpha,\beta) \in J^{(k)}; \end{cases}$
4) if $u^{(k)} \neq \mathbf{0}$
    4.1) determine a suitable step length $\rho_k$;
    4.2) move to the next point using the relation $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \rho_k \boldsymbol{u}^{(k)}$;
    4.3) $k := k + 1$;
    4.4) goto 2);
5) else
    5.1) if $\partial E(\boldsymbol{r}^{(k)})/\partial r_{d\alpha\beta} \geq 0 \ \forall (d,\alpha,\beta) \in J^{(k)}$ EXIT;
    5.2) else
        5.2.1) delete from $J^{(k)}$ the index corresponding to the most negative value;
        5.2.2) goto 3);
```

One possible choice for the initial point $\boldsymbol{r}^{(0)}$ can be to derive it randomly but, more interestingly, it can be determined using some "traditional" statistical measure, combining the statistical, classical approach with the new optimization one.

The step length ρ_k can be determined by means of any line search technique [25] so that the function $\varphi(\rho) = E(\boldsymbol{r}^{(k)} - \rho \boldsymbol{u}^{(k)})$ is minimized, and $\boldsymbol{r}^{(k)} - \rho \boldsymbol{u}^{(k)}$ remains feasible. This usually requires evaluating the derivative $\varphi'(\rho)$, which should be computed in an iterative fashion as for partial derivatives. An alternative, less expensive approach is to use fixed step lengths, according to feasibility constraints. More precisely,

$$\rho_k = \min\{\rho^{(1)}, \rho^{(2)}_k\}, \tag{17}$$

where $\rho^{(1)}$ is some predetermined maximum step length, and

$$\rho^{(2)}_k = \min\left\{ \frac{r^{(k)}_{d\alpha\beta}}{u^{(k)}_{d\alpha\beta}} : u^{(k)}_{d\alpha\beta} > 0 \right\}. \tag{18}$$

The maximum step size $\rho^{(1)}$ should be carefully chosen. In fact, if $\rho^{(1)}$ is too small the algorithm converges slowly, while if it is too large an unexpected increase of E, or an oscillation about the minimum, can occur. An appropriate strategy could be to decrease $\rho^{(1)}$ slowly as the process converges.

A. Recursive Gradient Calculation

The algorithm described above requires evaluating partial derivatives of the objective function E:

$$\frac{\partial E(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \sum_{\gamma=1}^{N} \frac{\partial E_\gamma(\boldsymbol{r})}{\partial r_{d\alpha\beta}}. \tag{19}$$

In the case of quadratic error function $E^{(Q)}$ we have:

$$\frac{\partial E^{(Q)}_\gamma(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \sum_{i=1}^{n_\gamma} \sum_{\lambda=1}^{m} \left(p^{(F_\gamma)}_{i\lambda}(\boldsymbol{r}) - p^{(L_\gamma)}_{i\lambda}\right) \frac{\partial p^{(F_\gamma)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \tag{20}$$

while the logarithmic cost function $E^{(I)}$ yields

$$\frac{\partial E^{(I)}_\gamma(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = -\sum_{i=1}^{n_\gamma} \frac{\partial p^{(F_\gamma)}_{i\lambda^\gamma_i}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \left(p^{(F_\gamma)}_{i\lambda^\gamma_i}(\boldsymbol{r})\right)^{-1}. \tag{21}$$

Whatever the error function used, partial derivatives of the final labeling $\boldsymbol{p}^{(F_\gamma)}$ are to be computed. The final labeling of a relaxation process is the result of a number of iterations and depends on the stopping criterion used. This makes the analytical formulation of derivatives very difficult or even impossible when the number of iterations is not determined in advance. However, we have overcome the problem by developing some recursive formulas that will permit us to calculate derivatives with an iterative algorithm. For notational simplicity we put $h^{(t)}_{i\lambda}(\boldsymbol{r}) = p^{(t)}_{i\lambda}(\boldsymbol{r}) q^{(t)}_{i\lambda}(\boldsymbol{r})$, and suppress the index γ. We have

$$\frac{\partial p^{(t+1)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \frac{\frac{\partial h^{(t)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \sum_{\mu=1}^{m} h^{(t)}_{i\mu}(\boldsymbol{r}) - h^{(t)}_{i\lambda}(\boldsymbol{r}) \sum_{\mu=1}^{m} \frac{\partial h^{(t)}_{i\mu}(\boldsymbol{r})}{\partial r_{d\alpha\beta}}}{\left(\sum_{\mu=1}^{m} h^{(t)}_{i\mu}(\boldsymbol{r})\right)^2}, \tag{22}$$

where

$$\frac{\partial h^{(t)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = p^{(t)}_{i\lambda}(\boldsymbol{r}) \frac{\partial q^{(t)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} + \frac{\partial p^{(t)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} q^{(t)}_{i\lambda}(\boldsymbol{r}) \tag{23}$$

and

$$\begin{aligned} \frac{\partial q^{(t)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} &= \sum_{\delta\in\Delta_i} \sum_{\mu=1}^{m} \Bigg(\Phi(d=\delta \wedge \alpha=\lambda \wedge \beta=\mu) \\ &\quad \cdot p^{(t)}_{i+\delta,\mu}(\boldsymbol{r}) + r_{\delta\lambda\mu} \frac{\partial p^{(t)}_{i+\delta,\mu}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \Bigg) \\ &= \Phi(d\in\Delta_i \wedge \alpha=\lambda) p^{(t)}_{i+d,\beta}(\boldsymbol{r}) \\ &\quad + \sum_{\delta\in\Delta_i} \sum_{\mu=1}^{m} r_{\delta\lambda\mu} \frac{\partial p^{(t)}_{i+\delta,\mu}(\boldsymbol{r})}{\partial r_{d\alpha\beta}}. \end{aligned} \tag{24}$$

Here, Φ denotes a function that takes a Boolean expression X, and returns 0 or 1 according to its truth value:

$$\Phi(X) = \begin{cases} 0, & \text{if } X \text{ is false;} \\ 1, & \text{if } X \text{ is true.} \end{cases}$$

Notice that the initial probability labeling $\boldsymbol{p}^{(0)}$ does not depend on $\boldsymbol{r}$, and hence

$$\frac{\partial p^{(0)}_{i\lambda}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = 0, \qquad \text{for all } i,\ d,\ \lambda,\ \alpha,\ \beta. \tag{25}$$

Accordingly, (24) simplifies to

$$\frac{\partial q_{i\lambda}^{(0)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \Phi(d \in \Delta_i \wedge \alpha = \lambda) p_{i+d,\beta}^{(0)} \tag{26}$$

and consequently

$$\frac{\partial p_{i\lambda}^{(1)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \Phi(d \in \Delta_i) p_{i\lambda}^{(0)} p_{i+d,\beta}^{(0)} \cdot \frac{\Phi(\alpha = \lambda)\sum_{\mu=1}^{m} h_{i\mu}^{(0)}(\boldsymbol{r}) - p_{i\alpha}^{(0)} q_{i\lambda}^{(0)}(\boldsymbol{r})}{\left(\sum_{\mu=1}^{m} h_{i\mu}^{(0)}(\boldsymbol{r})\right)^2}. \tag{27}$$

Equations (22)–(24) define a discrete-time dynamical system, with initial conditions given by (25), that can be used for evaluating the derivatives needed to the learning algorithm. The process begins at time step $t = 0$ and evolves in parallel with relaxation labeling. To compute exactly the derivatives of the final labelings $\boldsymbol{p}^{(F)}$'s, the process must be stopped when relaxation labeling terminates, according to its own stopping criterion. Thus, if relaxation stops at step $t = T$, the dynamical system will terminate at time step $t = T$, too.

B. Computational Requirements and Numerical Stability

It is clear that the major computational effort of the algorithm stems from the evaluation of partial derivatives. In fact, a number of derivatives proportional to m^3 is needed to maintain and update at each iteration (m being the number of labels); moreover, the algorithm takes a time roughly proportional to m to compute each derivative, making at all $O(m^4)$ calculations on each step. Nevertheless, the computational complexity of the algorithm seems not to be a severe limitation because in most interesting applications of relaxation labeling the number of labels is not so large as to preclude the applicability of the method. In addition, it is easily seen that computing each derivative $\partial p_{i\lambda}/\partial r_{d\alpha\beta}$ requires local information only, making the algorithm especially suited for parallel implementation. However, updating each compatibility strength needs access to nonlocal information, as seen from (20) or (21).

In order to successfully implement the algorithm, some practical considerations about its numerical stability are required, which are especially useful when "sparse" labelings (i.e., containing many zeros) are involved. In the course of the optimization process, in fact, it might well happen that some compatibility component $r_{\delta\lambda\mu}$ becomes zero. In case of sparse labelings, this could result in a "divide-by-zero" error, when formulas (1), (22), and (27) are evaluated. To circumvent this difficulty, problem (10) can be rewritten as

$$\begin{aligned} &\text{minimize } E(\boldsymbol{r}), \boldsymbol{r} \in R^D \\ &\text{subject to } r_{d\alpha\beta} \geq \epsilon, \end{aligned} \tag{28}$$

where, now, ϵ is a small positive constant. This new problem formulation involves a straightforward modification to the condition for the active constraints and the step size determination formula (18), and we will omit the details.

In addition, another subtle problem could arise if the $r_{d\alpha\beta}$'s are allowed to become too small. In fact, it is easily seen that partial derivatives could become too large or, even, an overflow condition could occur. Again, this is especially likely when sparse labelings are involved. In some special circumstances, and when fixed or adjustable step sizes are used, this situation can potentially result in unexpected jumps toward "poor" points. This actually occurred in our early experiments [37], [38]. In order to avoid these difficulties we found it convenient to keep the value of ϵ not excessively small.

V. Comparison with Neural Network Learning Algorithms

There exists an acknowledged connection between relaxation labeling and certain models of neural networks for which, in recent years, a number of learning algorithms have been developed (see [29] for an excellent and up-to-date introduction to this field). Thus it is instructive to discuss the analogies and the differences between our learning algorithm and those of neural networks, and point out the implications of the proposed approach.

At first, the analogy between relaxation labeling and neural networks appears evident: both are parallel processing models consisting of highly interconnected units that perform simple local computations. They make use of distributed knowledge representation which is encoded in the connection strengths between processing elements (the compatibility coefficients in relaxation labeling). In particular, relaxation labeling can be regarded as a recurrent network (with no hidden units) and, in this respect, is much similar to the model developed by Hopfield [9], [10]. Moreover, this similarity is further strengthened by the property that both the models have a common Liapunov (or energy) function, when the weight (compatibility) matrix is symmetric [7], [9], [10].

However, a little more detailed analysis reveals that the two models differ significantly in at least two ways. Firstly, the task carried out by the relaxation labeling processing units is much more complicated because relaxation (unlike the Hopfield net) performs a mapping in probability space. This implies that some form of projection is needed in relaxation labeling to accommodate constraints: in the simplest nonlinear scheme of Rosenfeld *et al.* [3] this is accomplished by a straightforward row normalization, but much more complex projection operators can be devised [8], [39], [40]. The inability of neural networks to deal with constraints is usually overcome by adding appropriate penalty terms into the energy function so as to "encourage" valid solutions [41]. Unfortunately, experimental analyses revealed that this approach is far from being reliable because the resulting computed solutions frequently violate many of the imposed constraints [42], [43]. The second major difference between relaxation labeling and neural networks is perhaps more important and concerns their dynamic collective behavior. It has been previously stated that the symmetry of the weight (compatibility) matrix assures the existence of a common energy function which is spontaneously minimized as the processes evolve. This implies that the two dynamical systems will always converge toward stable configurations. A fundamental result of Hummel and Zucker [7] assures that relaxation labeling still performs useful computations even if the symmetry condition is removed: they proved that relaxation processes are attracted by strictly consistent labelings (which lie on the corners of the search space) whether or not the compatibility matrix is symmetric. Interestingly, the same result, though demonstrated for a mathematically-derived class of relaxation processes, remains valid for the simple heuristic nonlinear scheme of formula (1) [44]. As Hummel and Zucker pointed out, their local convergence result is much more desirable than a global convergence result as the final solutions depend on the initial configurations. By contrast, when the connections are allowed to be asymmetric, the Hopfield network is no longer guaranteed to converge toward stable states and it can exhibit a cyclic or a chaotic behavior [9], [10]. It should be mentioned that the symmetry requirement is recognized to be one of the major drawbacks of the Hopfield model since makes it unplausible as a biological model.

Now, let us focus on the learning problem. Learning in recurrent neural networks has proved to be much more difficult than in feed-forward architectures, for which the well-known back-propagation algorithm has been developed [45]. Various generalizations of back-propagation for arbitrary recurrent networks have been proposed [46]–[48], which are entirely based on the assumption that the network being trained settles down to a stable state. Recurrent back-propagation (like its original feed-forward version) is basically a two-step procedure: the first step involves relaxing the network to reach a stable state, while the second propagates the discrepancy between the desired and the actual output values backwards, and updates the connection strengths accordingly. In contrast with recurrent back-propagation, our learning algorithm is extremely general because it does not make any assumption about the dynamic behavior of the relaxation process and, to work, it is not required to wait for relaxation to converge. Rather, as described previously, a given (arbitrary) criterion is used to stop the relaxation process (as well as the calculation of derivatives) whether or not a stable state has been reached. Moreover, unlike back-propagation, it does not make use of a successive backward pass for computing derivatives but it propagates them forwards in parallel with the relaxation labeling process. In the field of neural networks the idea of propagating derivatives forwards has been used by Williams and Zipser [49] in the very different context of learning temporal sequences. There, convergence to a stable attractor is not desired; instead, a network is trained to go through a predetermined sequence of states (a limit cycle) in response to specific input stimuli. Recurrent networks of this kind appear to be very promising to solve problems like speech recognition and grammatical inference, where temporal dependencies are to be captured. Unfortunately, current learning algorithms for such networks are computationally very expensive and this has seriously limited their applications to real-world problems of practical interest. Observe that relaxation labeling is not suited for such a task due to its convergence properties.

As a final remark, we would like to emphasize that recently much attention has been devoted to the problem of establishing a formal correspondence between the fields of relaxation labeling and neural networks [50], [51]. In particular, Yu and Tsai [51] have stressed that one resulting advantage of strengthening such a relationship is that research results of one domain can be instructive to the other. The work described in this paper can be considered as a step toward such a direction: the idea of learning, standard within the neural network community, has now been incorporated into the relaxation labeling domain, making the two fields more closely related; the resulting learning algorithm can be usefully employed in all the applications where relaxation labeling is being used as a standard technique.

VI. An Example: Part-of-Speech Disambiguation

In this section, we present some experiments that prove the effectiveness of the proposed approach. In particular, we tested our learning algorithm over a novel application of relaxation labeling that involves tagging words according to their correct parts-of-speech, as determined by the context in which they occur. This is a fundamental problem that arises not only in linguistics but also in various, more practical applications such as speech recognition, optical character recognition, and speech synthesis [52], [53], to name just a few.

In this application, the objects to be labeled are words of a sentence $\boldsymbol{W} = w_1 \cdots w_n$ and the labels are the parts-of-speech. The word labeling task can be accomplished by a two-step procedure. First, by means of some local analysis, one derives an initial labeling assignment $\boldsymbol{p}^{(0)}$. The simplest way of doing this consists of using a dictionary look-up which provides for each word the list of its potential parts-of-speech, but more sophisticated methods that exploit the orthographic structure of words have been developed [54]. Due to the presence of homographs in natural language, that is words belonging to more than one syntactic class, local information does not suffice to achieve good labeling results; therefore, in the second step of the word-labeling procedure, contextual constraints are taken into account. This task can be accomplished by a relaxation labeling process [55], where the compatibility coefficients express the strength of agreement between neighboring syntactic classes.

In the experiments presented here, the initial labelings $\boldsymbol{p}^{(I)}$'s were constructed by uniformly distributing the probability mass among the labels found into a dictionary look-up. More precisely, let $\Lambda_i \subseteq \Lambda$ be the set of possible labels for word w_i, as found by consulting the dictionary; then

$$p_{i\lambda}^{(0)} = \begin{cases} 1/|\Lambda_i|, & \text{if } \lambda \in \Lambda_i, \\ 0, & \text{otherwise.} \end{cases}$$

The final labelings $\boldsymbol{p}^{(F)}$'s, instead, were obtained by stopping the relaxation process after the first iteration. The neighborhood chosen for disambiguation contained only the right offset position (i.e., $\Delta_i = \{+1\}$), while the label set Λ consisted of the main parts-of-speech: verb, noun, adjective, adverb, determiner, conjunction, preposition, pronoun, plus a special miscellaneous label.

In the first phase of our experiments, we took a 3,500-word sample text containing sentences extracted from some issues of the EEC Italian Official Journal. This was part of a larger corpus that was subject to a semi-automatic labeling within the ESPRIT Project 860 "Linguistic analysis of the European languages" [56]. We divided the sample text into three separate parts. The first one (containing about 1,500 words) was used to derive two different statistical compatibility vectors to be used as the initial points for the learning algorithm. More specifically, we determined correlation-based coefficients

$$r^{(0)}_{\delta\lambda\mu} = 1 + \frac{P_\delta(\lambda, \mu) - P(\lambda)P(\mu)}{\sqrt{(P(\lambda) - P(\lambda)^2)(P(\mu) - P(\mu)^2)}} \tag{29}$$

and Peleg's compatibilities [18]

$$r^{(0)}_{\delta\lambda\mu} = \frac{P_\delta(\lambda, \mu)}{P(\lambda)P(\mu)}. \tag{30}$$

Here $P_\delta(\lambda, \mu)$ denotes the probability that the pair (λ, μ) occurs, μ being at offset δ from λ, and $P(\cdot)$ is the marginal distribution of labels. Clearly, marginal probabilities can be derived from the joint distribution using the relation $P(\lambda) = \Sigma_\mu P_\delta(\lambda, \mu)$, for some δ.

Robust estimation of probabilities from naturally-occurred texts is acknowledged to be a difficult task due to the "sparseness" of real data [57]. Typically, the major problem is to allocate probabilities to never-occurred pairs, for which the standard maximum-likelihood estimate becomes zero. This problem is especially undesirable in our application in light of the considerations about numerical stability made in Section IV. One way to overcome the difficulty with maximum-likelihood estimation was suggested by A. M. Turing, and his idea was developed later by I. J. Good [58]. Turing's suggestion is to estimate the probability of a pair that occurred exactly r times in a sample of size N by r^*/N, where $r^* = (r + 1)n_{r+1}/n_r$ and n_r represents the frequency of the frequency r $(r \geq 0)$, that is the number of different pairs that occurred in the sample exactly r times. In the experiments reported here, we used a hybrid approach which combines both Turing's and maximum-likelihood estimates. The basic idea was suggested by Katz [57] in the context of estimating n-gram probabilities for the language models of speech recognition systems. Essentially, the idea is to estimate the probability of never-occurred pairs by means of Turing's formula, and the probability of pairs occurred more than, say, K times using the usual maximum-likelihood method, considering it as reliable. Finally, pairs occurred less than K times were given an estimate proportional to their counts, using a suitable smoothing coefficient derived so that the axiomatic properties of probabilities are fulfilled. The reader is referred to [38] for a more detailed discussion on this topic as well as the full derivation of estimation formulas. The results presented here were obtained by setting $K = 5$, as suggested in [57].

In addition to determining statistical-based starting points, a third initial point for the proposed learning algorithm was derived randomly.

Later, we took the second sample text and ran the proposed learning algorithm using a maximum step size $\rho^{(1)} = 0.1$, which appeared to be near-optimal. The text contained about 1,000 words (26 sentences), 148 of which were ambiguous (i.e., the dictionary provided more than one syntactic class). In Fig. 1 the behavior of the quadratic error function $E^{(Q)}$ is plotted for the three different starting points, along with the disambiguation accuracy[5] of relaxation during the learning process. Fig. 2, instead, shows learning curves for the logarithmic error function $E^{(I)}$. As we can see, after few iterations of the learning process the relaxation algorithm dramatically improved its performance, approaching nearly a 100% accuracy, regardless of the starting point.

TABLE I
DISAMBIGUATION ACCURACY OF RELAXATION LABELING OVER A 1,000-WORD TEST SAMPLE, USING BOTH THE INITIAL POINTS AND THE BEST POINTS FOUND BY THE LEARNING ALGORITHM

	Initial Points	Optimal Points	
		Quadratic Error	Logarithmic Error
Peleg	72.0%	88.2%	92.6%
Correlation	73.5%	93.4%	94.1%
Random	42.6%	89.7%	91.9%

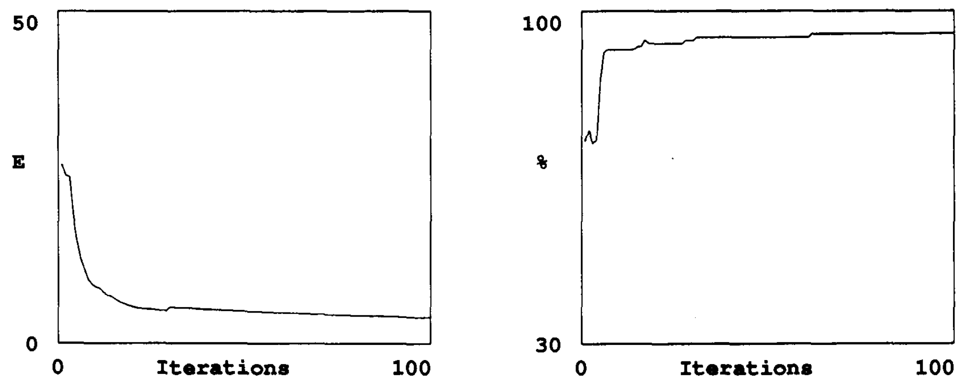
In order to assess the generalization ability of the algorithm, we tested the performance of relaxation labeling over the remaining 1,000-word test sample which contained 37 sentences and 136 ambiguous words. Table I makes a comparison among the disambiguation accuracies achieved by using the initial points and those obtained with the learning algorithm, both for the quadratic and logarithmic error functions. Each figure was computed with the best point produced by the algorithm, using the three above-mentioned starting points. As we can see, the learned compatibilities clearly outperform the standard statistical ones; moreover, the best generalization results were achieved using the logarithmic error function. It is interesting to observe that logarithmic error functions appear to produce better generalization results also in neural network learning [59].

In concluding this section, we mention that the word-class disambiguation task discussed here was tackled using a layered feed-forward neural network trained with the back-propagation algorithm [60]. Although a direct comparison with our results cannot be attempted especially because of the diversity of the languages used (English vs. Italian), we simply notice that they were able to achieve essentially our generalization performance (e.g., 94.7% accuracy) but using a much larger context window involving four preceding unambiguous words and one succeeding ambiguous word. In contrast, we achieved the same results by using only the succeeding ambiguous word. Clearly, a much more systematic comparison would be needed before drawing any significant conclusion.

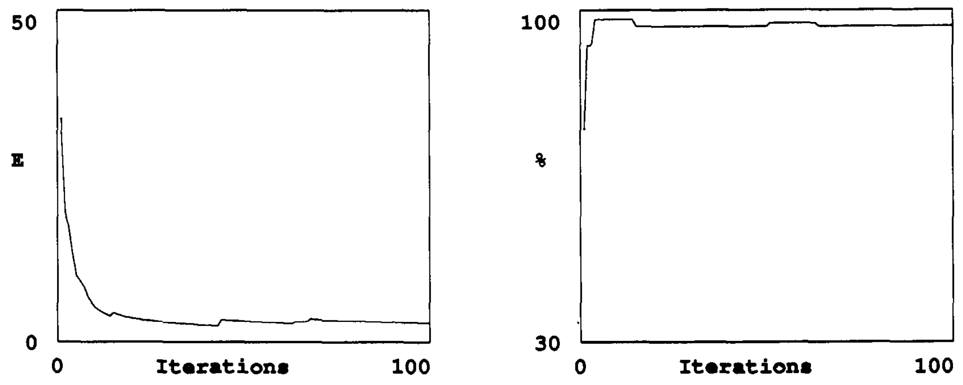
VII. SUMMARY AND CONCLUSION

In this paper, a new approach to determining the compatibility coefficients of relaxation labeling processes has been described. The viewpoint introduced here contrasts sharply with the traditional one since compatibility coefficients have

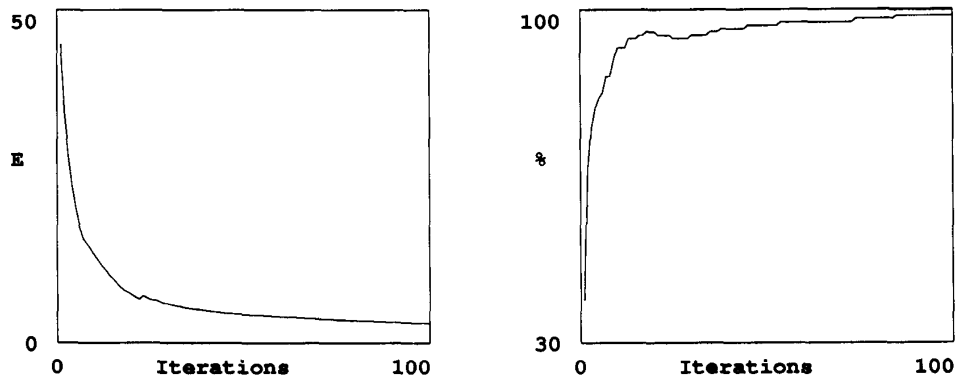
[5]The disambiguation accuracy is measured as the proportion of ambiguous words that are labeled correctly.



(a)



(b)



(c)

Fig. 1. Behavior of the quadratic error function (on the left side) and the corresponding disambiguation accuracy (on the right side) during the learning process, using different starting points: (a) Peleg's measure; (b) correlation; (c) random point.

been commonly interpreted on the basis of statistical or information-theoretic grounds. Our approach, instead, is based on rather "pragmatic" motivations: we derive them so that the performance of the relaxation algorithm over a sample of learning data is optimal. After formulating the problem of determining compatibilities in terms of an optimization problem, we have developed a straightforward algorithm for solving it based on the well-known gradient projection method of Rosen. One difficulty with the algorithm relies on the evaluation of the gradient of the objective function, as the iterative nature of the relaxation function makes the analytical formulation of derivatives prohibitive. In order to solve this problem, we have developed recursive formulas for derivatives which permit us to calculate them iteratively, in parallel with the relaxation process. Experiments over a practical application concerning syntactic category disambiguation have been presented, and the results achieved demonstrate that compatibility coefficients derived with the proposed optimization algorithm are clearly superior to standard statistical ones. The results are very encouraging as relaxation labeling demonstrated of being

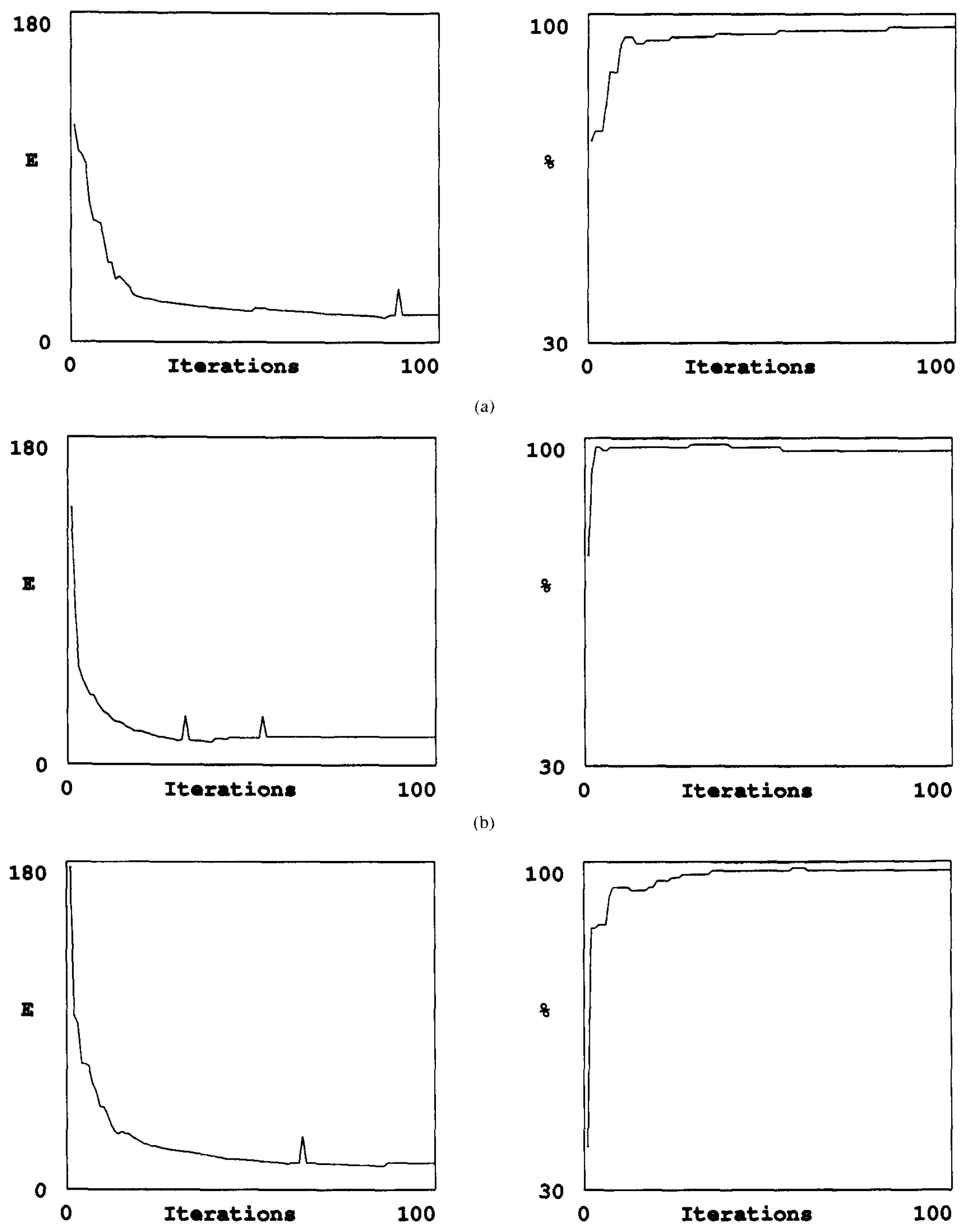


Fig. 2. Behavior of the logarithmic error function (on the left side) and the corresponding disambiguation accuracy (on the right side) during the learning process, using different starting points: (a) Peleg's measure; (b) correlation; (c) random point.

able to learn the constraints between Italian syntactic classes very quickly and using very limited contextual information.

In addition, a comparison with neural network learning procedures has been outlined, and it is argued that our algorithm can contribute to make the domains of relaxation labeling and neural networks more closely related. However, in spite of the close similarity between the two models, many questions remain to be answered in order for neural networks to be considered a practical technique for problems where relaxation labeling largely succeeds. As pointed out by Jamison and Schalkoff [61], the most serious problem to be tackled concerns the mapping of an arbitrary problem into the proper network architecture. Recently, this point has been addressed by Yu and Tsai [51].

The algorithm presented in this paper has mainly a practical value. Relaxation labeling is a standard technique in computer vision and pattern recognition, and is still receiving much attention by many researchers in these fields. The proposed algorithm is hoped to provide a definite answer to the old open question concerning the determination of compatibilities in real-world applications of relaxation labeling. It could be usefully employed, for example, in optical character recognition, automatic phonetic transcription (for which the celebrated NETtalk was developed [62]), multispectral pixel classification, and any other problem for which learning data are available.

Of course, several modifications and improvements to the basic algorithm presented here could be brought in, such as including second order information, choosing optimal step lengths, speeding up the convergence, and so on. Alternatively, global optimization techniques can be used [63], [64], which should prevent the algorithm from being trapped into poor local minima. However, it should be stressed that the main contribution of this paper is not the development of an algorithm for determining compatibility coefficients but, rather, the introduction of a new, unconventional standpoint from which they can be derived.

Our concluding considerations are rather speculative. It is generally acknowledged that a certain similarity exists between relaxation labeling and the structure of the human nervous system [7], [65], [66]. This is especially supported by the evidence that cerebral cortex contains extensive collateral interconnections among pyramidal neurons, and this strongly resembles the type of connections existing in a relaxation labeling network. Recently, Zucker *et al.* [67], have explicitly hypothesized that the first 2–3 iterations of relaxation labeling are carried out by the pyramidal cells connecting two areas of the visual cortex. From a biological point of view, the development of a learning mechanism for relaxation labeling has two major consequences. On the one hand, this removes one of the biggest obstacles to considering relaxation labeling as a plausible biological model. Human brain, in fact, learns from experience while relaxation labeling did not. On the other hand, assuming that relaxation labeling is actually implemented in the brain, one might wonder whether our learning algorithm has a biological plausibility. At this stage of development we can just note that some of the properties that biological learning mechanisms are supposed to have are fulfilled, but some others are not. For example, there is much experimental evidence in support of the hypothesis that biological learning schemes are error correcting, in the sense that learning is driven by the difference between what is required and what actually exists [68]. Our algorithm is clearly of this kind as the adjustment of a compatibility strength depends on the difference between the teacher signals and the actual outputs. Moreover, the algorithm overcomes the recognized implausibility of back-propagation which requires propagating information in reverse direction [69]. On the other hand, the proposed learning algorithm updates the compatibility strengths using nonlocal information and this seems to be the most serious argument against its biological plausibility.

However, whatever the value of such speculations may be, we believe that the development of suitable learning mechanisms, like the one proposed in this paper, can greatly enhance the potential of relaxation labeling in many practical applications. It is hoped that further experiments can serve to support our claims, and this is what we are currently trying to do.

APPENDIX
LEARNING SYMMETRIC COMPATIBILITIES

As discussed in Section V, the use of symmetric compatibilities guarantees the existence of an energy function that is minimized as the relaxation process evolves. This property has been shown to (approximately) hold also for the heuristic nonlinear relaxation scheme discussed in this paper [26]. Therefore, when symmetric coefficients are used we *do* know what task the relaxation process accomplishes, and it could be desirable that the learned coefficients be symmetric. In this appendix, we develop a straightforward modification of our learning algorithm that allows us to derive symmetric compatibility coefficients. In our notation, the symmetry condition is expressed by the relation $r_{\delta\lambda\mu} = r_{-\delta\mu\lambda}$, provided that $\delta \in \Delta_i \Rightarrow -\delta \in \Delta_{i+\delta}$. Observe that this last condition implies that Δ is symmetric, that is if $\delta \in \Delta$ then $-\delta \in \Delta$.

In order to derive symmetric compatibilities, problem (10) is to be reformulated as

$$\begin{aligned} &\text{minimize}\, E(\boldsymbol{r}), \boldsymbol{r} \in R^D \\ &\text{subject to} \\ &\qquad r_{d\alpha\beta} \geq 0 \\ &\qquad r_{d\alpha\beta} = r_{-d\beta\alpha}. \end{aligned} \tag{31}$$

Now, equality constraints have been introduced, and some modifications to the basic algorithm are required. Actually, the major difference with respect to Algorithm 1, relies on the determination of the direction $\boldsymbol{u}$ and the vector $\boldsymbol{\eta}$.

As in Section IV, let A denote the matrix of active constraints (we omit the subscript k, for simplicity). If, at a feasible point, s inequality constraints are active, $q = s/2$ of them will be redundant, because of equality constraints. For example, if $r_{d\alpha\beta} = 0$ then, necessarily, $r_{-d\beta\alpha} = 0$. In order for AA^T to be nonsingular, the rows of A must be linearly independent [24], and this means that q redundant inequality constraints must be dropped from A. Assuming that indices have been arranged appropriately, the constraint matrix A can be partitioned as

$$A = \begin{bmatrix} I & -I \\ C & O \end{bmatrix}, \tag{32}$$

where I is the $D/2 \times D/2$ identity matrix, O is the $q \times D/2$ null matrix, and C is a $q \times D/2$ matrix corresponding to the "first" inequality constraints, that is $C = [\boldsymbol{e}_{i_1}, \boldsymbol{e}_{i_2} \cdots \boldsymbol{e}_{i_q}]^T$, the $\boldsymbol{e}_j$'s being the standard unit vectors of $R^{D/2}$. We have

$$AA^T = \begin{bmatrix} 2I & C^T \\ C & I \end{bmatrix}, \tag{33}$$

where the number of rows and columns of the I's is clear from context. Simple calculations yield

$$(AA^T)^{-1} = \begin{bmatrix} X & -C^T \\ -C & 2I \end{bmatrix}, \tag{34}$$

where X is a $D/2 \times D/2$ diagonal matrix of the form $X = \operatorname{diag}(x_{11}, x_{22}, \cdots, x_{D/2,D/2})$, with

$$x_{ii} = \begin{cases} 1, & \text{if } i \text{ is the index of some active inequality constraint;} \\ 1/2, & \text{otherwise.} \end{cases}$$

Notice that $C^T C = 2X - I$ and, accordingly, we obtain

$$A^T(AA^T)^{-1}A = \begin{bmatrix} X & I-X \\ I-X & X \end{bmatrix}. \quad (35)$$

Finally, the projection matrix M is as follows:

$$M = I - A^T(AA^T)^{-1}A = \begin{bmatrix} I-X & I-X \\ I-X & I-X \end{bmatrix}. \quad (36)$$

The components of $\boldsymbol{\eta} = (AA^T)^{-1}A\nabla E(\boldsymbol{r})$ corresponding to the active inequality constraints are of the form $\eta_j = (\boldsymbol{e}_{i_j}^T, \boldsymbol{e}_{i_j}^T)\nabla E(\boldsymbol{r})$, for $j = 1 \cdots q$. In conclusion, in order to derive symmetric compatibilities, steps 3) and 5) of Algorithm 1 must be replaced with:

3) evaluate the vector $\boldsymbol{u}^{(k)}$, as follows:

$$u_{d\alpha\beta}^{(k)} = \begin{cases} \frac{1}{2}\left(\frac{\partial E(\boldsymbol{r}^{(k)})}{\partial r_{d\alpha\beta}} + \frac{\partial E(\boldsymbol{r}^{(k)})}{\partial r_{-d\beta\alpha}}\right), & \text{if } (d,\alpha,\beta) \notin J^{(k)}, \\ 0, & \text{if } (d,\alpha,\beta) \in J^{(k)}, \end{cases}$$

and

```
5) else
    1) if $\partial E(\boldsymbol{r}^{(k)})/\partial r_{d\alpha\beta} + \partial E(\boldsymbol{r}^{(k)})/\partial r_{-d\beta\alpha} \geq 0$
           $\forall (d,\alpha,\beta) \in J^{(k)}$ EXIT;
    2) else
        5.2.1) delete from $J^{(k)}$ the indices $(d,\alpha,\beta)$ and
               $(-d,\beta,\alpha)$ corresponding to the most negative
               value;
        5.2.2) goto 3).
```

ACKNOWLEDGMENT

The authors would like to thank R. Schiavo for reading early versions of the paper, and the anonymous referees for their helpful comments and suggestions.

REFERENCES

[1] R. M. Haralick and L. G. Shapiro, "The consistent labeling problem: Part I," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, no. 2, pp. 173–184, 1979.

[2] R. M. Haralick, L. S. Davis, A. Rosenfeld, and D. L. Milgram, "Reduction operations for constraint satisfaction," *Inform. Sci.*, vol. 14, pp. 199–219, 1978.

[3] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 6, pp. 420–433, 1976.

[4] S. Peleg and A. Rosenfeld, "Breaking substitution ciphers using a relaxation algorithm," *Commun. ACM*, vol. 22, no. 11, pp. 598–605, 1979.

[5] S. Peleg, "Ambiguity reduction in handwriting with ambiguous segmentation and uncertain interpretation," *Comput. Graph. Image Processing*, vol. 10, pp. 235–245, 1979.

[6] J. O. Eklundh and A. Rosenfeld, "Some relaxation experiments using triples of pixels," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 3, pp. 150–153, 1980.

[7] R. A. Hummel and S. W. Zucker, "On the foundations of relaxation labeling processes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 3, pp. 267–287, 1983.

[8] O. D. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, no. 4, pp. 412–424, 1981.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[10] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088–3092, 1984.

[11] S. E. Fahlman, G. E. Hinton, and T. J. Sejnowski, "Massively parallel architectures for AI: NETL, Thistle, and Boltzmann machines," in *Proc. Nat. Conf. Artificial Intell. (AAAI-83)*, Washington, DC, 1983, pp. 109–113.

[12] M. Kamada, K. Toraichi, R. Mori, K. Yamamoto, and H. Yamada, "A parallel architecture for relaxation operations," *Pattern Recognit.*, vol. 21, no. 2, pp. 175–181, 1988.

[13] Z. Chen, S. Lin, and Y. Chen, "A parallel architecture for probabilistic relaxation operations on images," *Pattern Recognit.*, vol. 23, no. 6, pp. 637–645, 1990.

[14] R. M. Haralick, J. L. Mohammed, and S. W. Zucker, "Compatibilities and the fixed points of arithmetic relaxation processes," *Comput. Graph. Image Processing*, vol. 13, pp. 242–256, 1980.

[15] D. P. O'Leary and S. Peleg, "Analysis of relaxation processes: The two-node two-label case," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 4, pp. 618–623, 1983.

[16] S. Peleg and A. Rosenfeld, "Determining compatibility coefficients for curve enhancement relaxation processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, no. 7, pp. 548–555, 1978.

[17] H. Yamamoto, "A method of deriving compatibility coefficients for relaxation operators," *Comput. Graph. Image Processing*, vol. 10, pp. 256–271, 1979.

[18] S. Peleg, "A new probabilistic relaxation scheme," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, no. 4, pp. 362–369, 1980.

[19] L. S. Davis and A. Rosenfeld, "Cooperating processes for low-level vision: A survey," *Artificial Intell.*, vol. 17, pp. 245–263, 1981.

[20] J. Kittler, "Compatibility and support functions in probabilistic relaxation," in *Proc. 8th Int. Conf. Pattern Recognit.*, Paris, France, 1986, pp. 186–189.

[21] R. A. Hummel, "A design method for relaxation labeling applications," in *Proc. Nat. Conf. Artificial Intell. (AAAI-83)*, Washington, DC, 1983, pp. 168–171.

[22] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. Comput.*, vol. C-26, no. 4, pp. 394–403, 1977.

[23] L. S. Davis and A. Rosenfeld, "Curve segmentation by relaxation labeling," *IEEE Trans. Comput.*, vol. C-26, no. 10, pp. 1053–1057, 1977.

[24] J. B. Rosen, "The gradient projection method for nonlinear programming—Part I: Linear constraints," *J. Soc. Indust. Appl. Math.*, vol. 8, no. 1, pp. 181–217, 1960.

[25] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.

[26] S. A. Lloyd, "An optimization approach to relaxation labeling algorithms," *Image Vision Comput.*, vol. 1, no. 2, pp. 85–91, 1983.

[27] J. Illingworth and J. Kittler, "Optimization algorithms in probabilistic relaxation labeling," in *Pattern Recognition Theory and Applications*, P. A. Devijver and J. Kittler, Eds. Berlin: Springer, 1987, pp. 109–117.

[28] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum-likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 2, pp. 179–190, 1983.

[29] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.

[30] J. Kittler and J. Illingworth, "Relaxation labeling algorithms—A review," *Image Vision Comput.*, vol. 3, no. 4, pp. 206–216, 1985.

[31] X. Zhuang, R. M. Haralick, and H. Joo, "A simplex-like algorithm for the relaxation labeling process," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 12, pp. 1316–1321, 1989.

[32] S. W. Zucker, Y. G. Leclerc, and J. L. Mohammed, "Continuous relaxation and local maxima selection: Conditions for equivalence," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, no. 3, pp. 117–127, 1981.

[33] S. Kullback, *Information Theory and Statistics*. New York: Wiley, 1959.

[34] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing, Vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 283–317.

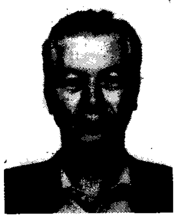
[35] S. A. Solla, E. Levin, and M. Fleisher, "Accelerated learning in layered neural networks," *Complex Syst.*, vol. 2, pp. 625–640, 1988.

[36] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Trans. Inform. Theory,* vol. IT-37, no. 1, pp. 145–151, 1991.
[37] M. Pelillo and M. Refice, "An optimization algorithm for determining the compatibility coefficients of relaxation labeling processes," in *Proc. 11th Int. Conf. Pattern Recognit.,* The Hague, The Netherlands, 1992, pp. 145–148.
[38] ——, "Learning compatibility coefficients for word-class disambiguation relaxation processes," in *Proc. Int. Conf. Spoken Language Processing,* Banff, Canada, 1992, pp. 389–392.
[39] J. L. Mohammed, R. A. Hummel, and S. W. Zucker, "A gradient projection algorithm for relaxation methods," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-5, no. 3, pp. 330–332, 1983.
[40] P. Parent and S. W. Zucker, "Radial projection: An efficient update rule for relaxation labeling," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 11, no. 8, pp. 886–889, 1989.
[41] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.,* vol. 52, pp. 141–152, 1985.
[42] G. V. Wilson and G. S. Pawley, "On the stability of the traveling salesman problem algorithm of Hopfield and Tank," *Biol. Cybern.,* vol. 58, pp. 63–70, 1988.
[43] G. W. Davis, "Sensitivity analysis in neural net solutions," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-19, no. 5, pp. 1078–1082, 1989.
[44] T. Elfving and J. O. Eklundh, "Some properties of stochastic labeling procedures," *Comput. Graph. Image Processing,* vol. 20, pp. 158–170, 1982.
[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing, Vol. 1: Foundations,* D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[46] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Phys. Rev. Lett.,* vol. 59, no. 19, pp. 2229–2232, 1987.
[47] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proc. Int. Conf. Neural Networks,* San Diego, CA, 1987, pp. 609–618.
[48] R. Rohwer and B. Forrest, "Training time-dependence in neural networks," in *Proc. Int. Conf. Neural Networks,* San Diego, CA, 1987, pp. 701–708.
[49] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computat.,* vol. 1, pp. 270–280, 1989.
[50] C. Torras, "Relaxation and neural learning: Points of convergence and divergence," *J. Parallel Distrib. Comput.,* vol. 6, pp. 217–244, 1989.
[51] S. S. Yu and W. H. Tsai, "Relaxation by the Hopfield neural network," *Pattern Recognit.,* vol. 25, no. 2, pp. 197–209, 1992.
[52] A. M. Derouault and B. Mérialdo, "Natural language modeling for phoneme-to-text transcription," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-8, no. 6, pp. 742–749, 1986.
[53] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *Proc. Int. Conf. Acoust., Speech, Signal Processing,* Glasgow, Scotland, 1989, pp. 695–698.
[54] M. Pelillo, F. Moro, and M. Refice, "Probabilistic prediction of parts-of-speech from word spelling using decision trees," in *Proc. Int. Conf. Spoken Language Processing,* Banff, Canada, 1992, pp. 1343–1346.
[55] M. Pelillo and M. Refice, "Syntactic category disambiguation through relaxation processes," in *Proc. EUROSPEECH 91,* Genova, Italy, 1991, pp. 757–760.
[56] L. Boves and M. Refice, "The linguistic processor in a multi-lingual text-to-speech and speech-to-text conversion system," in *Proc. Europ. Conf. Speech Technol.,* Edimburgh, Scotland, 1987, pp. 385–401.
[57] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-35, no. 3, pp. 400–401, 1987.
[58] I. J. Good, "The population frequencies of species and the estimation of population parameters," *Biometrika,* vol. 40, no. 3/4, pp. 237–264, 1953.
[59] M. J. J. Holt, "Comparison of generalization in multi-layer perceptrons with the log-likelihood and least-squares cost functions," in *Proc. 11th Int. Conf. Pattern Recognit.,* The Hague, The Netherlands, 1992, pp. 17–20.
[60] J. Benello, A. W. Mackie, and J. A. Anderson, "Syntatic category disambiguation with neural networks," *Comput. Speech Language,* vol. 3, pp. 203–217, 1989.
[61] T. A. Jamison and R. J. Schalkoff, "Image labeling: A neural network approach," *Image Vision Comput.,* vol. 6, no. 4, pp. 203–213, 1988.
[62] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Syst.,* vol. 1, pp. 145–168, 1987.
[63] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.
[64] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications.* Dordrecht: Reidel, 1987.
[65] D. H. Ballard, G. E. Hinton, and T. J. Sejnowski, "Parallel visual computation," *Nature,* vol. 306, pp. 21–26, 1983.
[66] J. A. Anderson, "Cognitive and psychological computation with neural models," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-13, no. 5, pp. 799–815, 1983.
[67] S. W. Zucker, A. Dobbins, and L. Iverson, "Two stages of curve detection suggest two styles of visual computation," *Neural Computat.,* vol. 1, pp. 68–81, 1989.
[68] G. Mitchinson, "Learning algorithms and networks of neurons," in *The Computing Neuron,* R. Durbin, C. Miall, and G. Mitchinson, Eds. Reading, MA: Addison-Wesley, 1989, pp. 35 53.
[69] F. Crick, "The recent excitement about neural networks," *Nature,* vol. 337, pp. 129–132, 1989.

Marcello Pelillo (M'92) was born in Taranto, Italy, on June 1, 1966. He received the degree with honors in computer science from the University of Bari, Italy, in 1989.

From 1988 to 1989, he spent about a year at the IBM Scientific Center in Rome, where he was involved in studies on natural language and speech recognition. In 1991, he became a Researcher at the Department of Computer Science of the University of Bari, Italy. His current research interests include neural networks, pattern recognition, and artificial intelligence.

Dr. Pelillo is a member of the IEEE Computer Society and the Pattern Recognition Society.



Mario Refice (M'89) was born in Rome, Italy, 1941. He received his "Laurea" in physics in 1967.

Since the beginning of his research activity, at the Physics Institute of Bari University, his interests have been oriented towards information and computer science. From 1970 to 1984, his efforts have been concentrated in setting and helping to grow the Center for Studies and Applications in Advanced Technologies, in Bari, aiming at fostering technological innovation in Southern Italy. In 1985, he became Associate Professor of Formal Languages and Compilers at the Institute of Information Science, Bari University. In 1989, he moved to the newly established Politecnico di Bari where he is currently Associate Professor of Foundations of Computer Science in the Department of Electrical Engineering and Electronics. His research interests currently focus on speech processing and computational linguistics. Having served for several Italian and European associations, including ADA-Europe, he is member of the ACM and the AICA.