Labeling Evaluation in Probabilistic Networks*

SHMUEL PELEG

Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, Maryland 20742

Communicated by Laveen Kanal

ABSTRACT

In many pattern recognition problems a probabilistic labeling of a network is given, and it is desired to obtain a unique unambiguous labeling for the network. This labeling should be influenced by the given probabilistic labeling, and by the joint distributions of the labels at subsets of nodes of the network. Relaxation algorithms have frequently been used to find such a labeling, but no method has been available to evaluate the results or to compare two different labelings. A measure is proposed here for evaluating labelings based on the given probabilistic labeling and joint distributions. This measure can also be used to define a termination criterion for relaxation by evaluating the labeling at each iteration. In addition, it could be used to evaluate labelings derived by any other process, and to guide heuristic search.

1. INTRODUCTION

Many pattern recognition tasks involve networks of elements, in which each element is to be classified (or labeled) subject to constraints associated with the arcs of the network. The constraints are expressed by a probabilistic model giving the joint distributions of labels at nodes connected by arcs. An initial classification is given by assigning a probability vector to each node, specifying the probability of the node having each label.

In this paper we propose a measure for evaluating labelings of a network in terms of the given initial probability estimates and a given set of *a priori* joint label probabilities. For example, let p_1 and p_2 be the probability vectors at two neighboring nodes, and let P_{12} be the *a priori* joint distribution of pairs of

*The support of the National Science Foundation under Grant MCS-76-23763 is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

52 Vanderbilt Ave., New York, NY 10017 0020-0255/80/06213-08$01.75

labels at those two nodes. We want an interpretation that maximizes some combination of p_1, p_2, and P_{12}. As in [4], it is proposed here that the product is a good combination. Thus a good labeling $\langle \alpha, \beta \rangle$ for the two nodes should maximize $p_1(\alpha)p_2(\beta)P_{12}(\alpha,\beta)$. The measure is also generalized for the case of a general network.

Once we have the goal of maximizing a specific measure, relaxation processes [6, 8, 9] can be tested at every iteration to see whether they improve that measure. The measure could also be used in a search algorithm for the maximum labeling, in cases where simple methods such as dynamic programming [4] are not feasible.

2. THE LIKELIHOOD MEASURE

Let $V = \{v_1, \ldots, v_n\}$ be a set of nodes, let $E^k \subseteq V^k$, $k \leqslant n$, be a set of arcs (or relations), and let Λ be a set of labels. The random variable l_i will designate the label associated with node v_i.

A *stochastic labeling* of the network assigns to each node v_i a probability vector $p_i : \Lambda \rightarrow [0, 1]$. $p_i(l_i = \alpha)$ is the probability of assigning label α to node v_i. The stochastic labeling is initially constructed using a classifier, which probabilistically classifies the objects that correspond to the nodes using their individual features.

A *probabilistic model* is a set of *a priori* joint probability distributions on E^k. For every $\langle v_1, \ldots, v_k \rangle \in E^k$, $P_{1,\ldots,k}(\alpha_1, \ldots, \alpha_k)$ designates the *a priori* probability of nodes $v_1, \ldots, v_k$ being labeled $\alpha_1, \ldots, \alpha_k$ respectively.

Every assignment of labels $\Omega = \alpha_1, \ldots, \alpha_n$ to $v_1, \ldots, v_n$ has two probabilities: the probability assigned to it by the stochastic labeling, $P_S(\Omega)$, and the probability assigned to it by the probabilistic model, $P_M(\Omega)$. In most cases, the initial probabilities are assigned by a local classifier that does not use any knowledge from the probabilistic model. In edge detection [10], for example, a local edge detector is used to assign the initial probabilistic labeling. The edge detector is based on local gray level differences only, and does not use any cooccurrence knowledge from the model. Since P_S and P_M are derived independently from different sources, the probability of the labeling Ω can be estimated by multiplying these two probabilities. Thus, we get

$$P(\Omega) = P_S(\Omega)P_M(\Omega). \tag{1}$$

The maximum likelihood labeling is the labeling that maximizes $P(\Omega)$. The computation of P_S and P_M will now be discussed. Since computing $P(\Omega)$ will involve a product of many probabilities, the product can be replaced by a sum of logarithms to prevent underflow. A maximal probability will, of course, produce a maximum logarithm.

2.1. THE PROBABILITY P_S

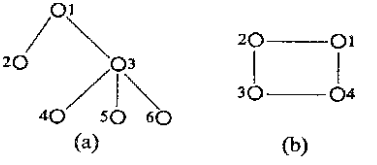
The probability P_S is based on the stochastic labeling of the network. Since the assignment of probabilities to each node v_i by the classifier is determined by examining the object corresponding to v_i only, we can assume independence of the assignments. Under this assumption, the probability that the network will have the labeling $\Omega = \alpha_1, \ldots, \alpha_n$ is the product of the individual probabilities $p_i(l_i = \alpha_i)$:

$$P_S(\Omega) = \prod_{i=1}^{n} p_i(l_i = \alpha_i). \tag{2}$$

2.2. THE PROBABILITY P_M

Computing the probability P_M, based on the probabilistic model, is more difficult. To find P_M, we need the joint probability distribution of all the nodes, but we know only the distributions for some subsets of nodes. Extending probability distributions from subsets is a known problem, and some methods for doing it are discussed in [1, 2, 3]. These methods can be applied to some networks, but are not practical for large and complicated networks.

Fig. 1. Graph dependencies: (a) tree dependency, (b) cyclic dependency.



The most convenient network for purposes of extension is a tree, under the assumption that the joint probabilities of every two nodes connected by an arc are given. For the dependency tree in Fig. 1(a), the maximum entropy extension to the *a priori* pairwise probabilities for $\Omega = \alpha_1, \ldots, \alpha_6$ is

$$P(l_1 = \alpha_1) P(l_2 = \alpha_2 | l_1 = \alpha_1) P(l_3 = \alpha_3 | l_1 = \alpha_1) P(l_4 = \alpha_4 | l_3 = \alpha_3)$$

$$\times P(l_5 = \alpha_5 | l_3 = \alpha_3) P(l_6 = \alpha_6 | l_3 = \alpha_3)$$

$$= \frac{\begin{array}{r} P(l_1 = \alpha_1, l_2 = \alpha_2) P(l_1 = \alpha_1, l_3 = \alpha_3) P(l_3 = \alpha_3, l_4 = \alpha_4) \\ \times P(l_3 = \alpha_3, l_5 = \alpha_5) P(l_3 = \alpha_3, l_6 = \alpha_6) \end{array}}{P(l_1 = \alpha_1) P(l_3 = \alpha_3)^3}.$$

$P(l_i=\alpha_i)$ is an *a priori* probability computed from the *a priori* joint probabilities as a marginal.

In general, it can be seen that for any tree $G=(V,E)$, the extended probability for $\Omega=\alpha_1,\ldots,\alpha_n$ will be

$$P_M(\Omega)=\prod_{(v_i,v_j)\in E}\frac{P(l_i=\alpha_i,l_j=\alpha_j)}{P(l_i=\alpha_i)P(l_j=\alpha_j)}\prod_{i=1}^{n}P(l_i=\alpha_i)$$

$$=\left[\prod_{(v_i,v_j)\in E}P(l_i=\alpha_i,l_j=\alpha_j)\right]\left[\prod_{i=1}^{n}P(l_i=\alpha_i)^{N_i-1}\right]^{-1}, \tag{3}$$

where N_i is the degree of node v_i. It was shown [3] that this extension is the maximum entropy extension having marginals which are identical to the given joint probabilities.

The expression (3) is correct only for tree dependencies. For other dependencies, such as the one in Fig. 1(b), it is not valid. In such a case, the marginals of the distribution obtained when (3) is used will differ from the given pairwise probabilities. However, an approximation can be found in such cases by using (3) for a spanning tree of the given network, as is done in [3]. Spanning trees of digital pictures on rectangular grids have been used in some work on stochastic tree grammars for pictures [11]. In this approach, we might retain all the column dependencies (each pixel has joint probabilities with the pixels above and below it), but row dependencies remain on only one row. Thus half of the given joint probabilities are not used. Better estimates that use all the joint probabilities are desirable.

Recent investigations on extending the partial joint distributions to a global estimate for networks [13, 14] have not yet led to computationally feasible methods for nontree networks. Thus, the examples given in the following sections will be chosen from domains where P_M can be computed on a tree or string structure.

3. TWO EXAMPLES FROM THE TEXT DOMAIN

In the text domain, the stochastic labeling is a sequence $Q=q_1\ldots q_n$ of probability vectors over an alphabet. $q_k(\lambda)$ represents the probability of the kth character being the letter λ. According to (2), the probability of a given string of letters $\Omega=\alpha_1\ldots\alpha_n$ based on the stochastic labeling is

$$P_S(\Omega)=\prod_{i=1}^{n}q_i(\alpha_i). \tag{4}$$

The probabilistic model for text will be based on the joint distributions of n-grams, i.e., n-tuples of letters in the appropriate language. These distributions can be extracted from a stochastic grammar that generates the language [12], or simply by frequency counting in a large sample of text. In this section trigrams will be used, as they were found to be very powerful in [5, 7].

Let $\text{Prob}(\alpha_1, \alpha_2, \alpha_3)$ be the *a priori* probability of a random trigram being $\alpha_1\alpha_2\alpha_3$, and let $\text{Prob}(\alpha_1, \alpha_2)$ be the *a priori* probability of a random digram being $\alpha_1\alpha_2$. Note that $\text{Prob}(\alpha_1, \alpha_2)$ is a marginal distribution of $\text{Prob}(\alpha_1, \alpha_2, \alpha_3)$.

In this case of a string, having trigram probabilities, the model probability P_M is defined recursively as follows: when the string is of length 3, the measure is simply

$$P_M(\alpha_1\alpha_2\alpha_3) = \text{Prob}(\alpha_1, \alpha_2, \alpha_3).$$

Assume we have a measure for a string of length N. Adding one more letter, the measure can be computed by

$$\begin{aligned} P_M(\alpha_1 \ldots \alpha_{N-1}\alpha_N\alpha_{N+1}) &= P_M(\alpha_1 \ldots \alpha_N)\,\text{Prob}(\alpha_{N+1}|\alpha_1 \ldots \alpha_N) \\ &= P_M(\alpha_1 \ldots \alpha_N)\,\text{Prob}(\alpha_{N+1}|\alpha_{N-1}\alpha_N) \\ &= P_M(\alpha_1 \ldots \alpha_N)\frac{\text{Prob}(\alpha_{N-1}\alpha_N\alpha_{N+1})}{\text{Prob}(\alpha_{N-1}\alpha_N)}. \end{aligned} \tag{5}$$

According to (1), the combined measure for a string $\Omega = \alpha_1 \ldots \alpha_n$ will be

$$P(\Omega) = P_S(\Omega)P_M(\Omega),$$

where P_S and P_M are as in (4) and (5) respectively. The usefulness of this measure is demonstrated in two examples.

The first example is based on [5], where ambiguously segmented handwriting was to be recognized. Three handwritten words were considered: LET, EVEN, and BOOK. By choosing different segmentations and different interpretations for the letters, many possible words can be obtained. There were 13 possible interpretations for LET, 848 for EVEN, and 3960 for BOOK. Table 1 shows merits for the five highest-merit interpretations in each case. The merit used in this table is directly proportional to the probability $P(\Omega)$. In all three cases, the correct word has the highest merit.

TABLE 1

The Five Interpretations Having Highest Merit for the Handwritten Words (a) LET, (b) EVEN, and (c) BOOK

	a		b		c	
	LET	506	EVEN	61	BOOK	66
	HT	73	BEEN	23	HOOK	53
	ELT	54	HEEN	15	BUSH	15
	ECT	37	EVERS	14	HUSH	11
	EST	34	BLEN	6	EMAK	10
Total number of possible words		13		848		3960

The second example involves breaking substitution ciphers [7]. A substitution cipher is a permutation of the alphabet; every letter in a given "plaintext" message is replaced by another letter to define a coded message. Given the coded message, any of the 26! possible permutations could be the plaintext message. Evaluating the merit for all possible cases, and choosing the permutation giving the highest merit, as was done in the handwriting case, is not practical.

A relaxation algorithm for finding the right permutation is discussed in [7]. Rather than choosing one permutation, the relaxation process attempts to iteratively improve the initial stochastic labeling. We construct a key from a stochastic labeling by choosing for every node (i.e., every code letter) the label (plaintext letter) having the highest probability. The key thus chosen is not necessarily a permutation, but in the experiments that were performed the key became closer to the correct permutation after each iteration. These experiments were done for a coded message having a known key. In the case of a message with an unknown key, a measure such as that suggested earlier can serve as an indication of the progress of the relaxation iterations, and can be used as a termination criterion.

Direct application of the measure P will not be helpful here. Since the measure is a product, one zero element will cause the entire product to be zero. It is very unlikely that any key except the correct one will satisfy all the language constraints, and all trigrams in the decoded message will have positive *a priori* probability. In the examples used in [7], even the correct keys had a zero measure, since the messages contained trigrams that did not appear in the reference text used to compute the trigram probabilities, and thus these trigrams had zero *a priori* probability. For example, the word TECHNIQUE had three trigrams, CHN, HNI, NIQ, that were given zero *a priori* probability. To handle such cases, instead of using the above measure, we can count the

number of trigrams having zero *a priori* probability, which is the number of zero elements in the product to compute $P(\Omega)$. Each zero corresponds to one inconsistency in the message obtained by using the suggested key, so that minimizing the number of zero elements minimizes the number of inconsistencies.

Table 2 displays, for the two examples of [7], the number of correct key entries and the number of zero elements for each iteration of the relaxation process. The number of zero elements decreases rapidly with the increase in the number of correct key entries. In one case, message 1, the relaxation process could not find the correct key entry for P. In this case, the last entry in the table is the number of zeros for the correct key, which is lower than at any stage of the relaxation. This suggests that a limited search after the termination of relaxation might find the correct key by searching for a key that produces a message with fewer zero probability trigrams.

TABLE 2

The Number of Zero Elements at Each Iteration of Relaxation for Two Coded Messages[a]

Message 1			Message 2		
Iteration	Correct entries in key	Number of zero elements in product	Iteration	Correct entries in key	Number of zero elements in product
0	5	337	0	8	393
1	9	301	1	13	215
2	14	143	2	17	152
3	16	48	3	18	120
4	17	42	4	20	105
5	19	12	5	21	79
6	20	10	6	23	67
7	21	9	7	23	67
8	21	9	8	24	28
9	21	9	9	24	28
10	22	8	10	25	26
11	22	8	11	25	26
			12	25	26
Correct			13	25	26
key	23	1	14	26	13

[a]For each message, the second column gives the number of correct entries in the suggested key, and the third column gives the number of zero elements in the product when this key is used. In the first message, the Gettysburg Address, only 23 letters occur. The relaxation process could not find the right entry for P, and the best result obtained was 22 correct key entries. The correct key, with 23 correct entries, is shown last for comparison.

4. CONCLUDING REMARKS

This paper suggests a measure for evaluating labelings of a network based on a probabilistic model and an initial stochastic labeling. This measure could serve as a criterion to establish whether labeling algorithms such as relaxation really give rise to improved labelings. It can also be used to terminate relaxation processes when the value of the measure begins to decrease. In cases where the best labeling obtained by the relaxation is not satisfactory, a limited search starting from that labeling might lead to an improved labeling.

REFERENCES

1. P. M. Lewis, Approximating probability distributions to reduce storage requirements, *Information and Control* 2:214–225 (1959).
2. D. T. Brown, A note on approximations to discrete probability distributions, *Information and Control* 2:386–392. (1959).
3. C. K. Chow and C. N. Liu, Approximating discrete probability distributions and dependency trees, *IEEE Trans. Information Theory* IT-14:462–467 (1968).
4. S. Peleg, Maximal derivations for probabilistic strings in stochastic languages, *Information and Control* 42:290–304 (1979).
5. S. Peleg, Ambiguity reduction in handwriting with ambiguous segmentation and uncertain interpretation, *Comput. Graphics and Image Processing* 10:235–245 (1979).
6. S. Peleg, A new probabilistic relaxation scheme, in *IEEE Conference on Pattern Recognition and Image Processing*, Chicago, Aug. 1979, pp. 337–343; *IEEE Trans. Pattern Analysis Machine Intelligence*, to appear.
7. S. Peleg and A. Rosenfeld, Breaking substitution ciphers using a relaxation algorithm, *Comm. ACM* 22:598–605 (1979).
8. A. Rosenfeld, Iterative methods in image analysis, *Pattern Recognition* 10:181–187 (1978).
9. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, Scene labeling by relaxation operations, *IEEE Trans. Systems, Man, and Cybernet.* SMC-6:420–433 (1976).
10. S. W. Zucker, R. A. Hummel, and A. Rosenfeld, An application of relaxation labeling to line and curve enhancement, *IEEE Trans. Computers* C-26:394–403 (1976).
11. S. Y. Lu and K. S. Fu, Stochastic tree grammar inference for texture synthesis and discrimination, *Computer Graphics and Image Processing* 9:234–245 (1979).
12. U. Grenander, *Pattern Analysis*, Lectures in Pattern Theory II, Springer, New York, 1978.
13. J. F. Lemmer, Algorithms for incompletely specified distributions in a generalized graph model for medical diagnosis, TR-505, Dept. of Computer Science, Univ. of Maryland, 1977.
14. J. F. Lemmer, A return to probabilities in computer assisted medical diagnosis, in *Proceedings of the International Conference on Cybernetics and Society*, 1977, pp. 612–616.

Received January 1980