Learning Compatibility Coefficients for Continuous Relaxation Labeling Processes

Chatavut Viriyasuthee

School of Computer Science

McGill University

ID: 260361503

April 24, 2010

Abstract

Relaxation labeling processes have been widely accepted in the past decades as one of the successful graph labeling algorithms. Given compatibilities between nodes in a graph, the processes converge to a certain labeling. However, if the compatibilities between nodes are not known, we may want to learn these relationships from a set of examples containing graph-labels pairs. In this project, we investigate the problems of Learning Compatibility Coefficients as presented by Pelillo in [1], where it treats the problems as optimization processes. Although it achieves a certain point, we extend the scope of the paper by identifying and modifying some unnecessary assumptions; and derive an optimization procedure according to this new formulation. We present experiment results which show the quality of the learned coefficients, in which the relative importances of compatibility within the given examples are captured.

Contents

Chapter 1

Introduction

Finding proper representations for given data is a type of problem that always involves in Machine Learning and Artificial Intelligence domain. Basically, these representations are important in terms of information interpretation for the higher level processors since they can reduce the dimension and ambiguities within data. However, they cannot be assigned freely because of the constraints that imposed by the relationships within information space. Relaxation Labeling is an algorithm that deals with the problems of finding proper representations, labeling, for given data and the constraints within them. For example, some applications of Relaxation Labeling can be found in [12], [8].

As we stated that in Relaxation Labeling, we want to find a proper labeling, called consistent labeling, for an input graph that satisfies its constraints, called compatibilities. There are two categories of Relaxation Labeling problems, discrete and continuous. In discrete cases, labeling is represented by predicates, while, in continuous cases, labeling is represented by probabilities. Both categories are defined on discrete work spaces and discrete label space [5]. There are other types of relaxation labeling problem where either work spaces, label spaces, or both are continuous; but they are beyond the scope of this work. In this work, we concern only the Continuous Relaxation Labeling problems where both work spaces and labeling are discrete.

All Continuous Relaxation Labeling problems can be viewed as nonlinear constrained optimization problems [10]. However, unlike most other optimization problems, people prefer the local optimum that corresponds to the initial labeling. Hummel proved that this local optimum is where the process reaches consistent labeling [5]. This is also indirectly equivalent to Haralick's definition of consistent labeling [3]. To reach the local optimum, many approaches derived from the theory of optimization have been presented. For example, we can also use approximation algorithms, such as one presented in [2], however these approaches may not be practical since

they consume high computation cost. In this case, since we want to reach the local optimum, the numerical steepest-ascend update rule can be used. The version of ascending algorithm for constrained optimization problem is to follow a projected gradient direction at each iteration [6]. However, Parent argued that this rule could sometime violate the constraints, so he presented another derivation of the gradient following; but instead of using normal projected gradient, he proposed the radial projection update rule [7].

The Relaxation Labeling requires that we know the compatibilities imposed by the relationships within data. On the other hand, if these relationships are unknown, we may want to discover them. In this case, for example, if a given domain has a corresponding complicate labeling, or we want to invent a system that can adapt to many labeling tasks, automatic compatibilities discovery is necessary. By giving a set of object-label pairs to train the process, this is undoubtedly Supervise Learning problems. There are many works based on this kind of application, [9]. In other words, since the Relaxation Labeling can be viewed as a version of recurrent Artificial Neural Network [4], the problem of compatibilities discovery can be viewed as ANN's training. In this work, we focuses on these problems, that is, how can we learn proper compatibility coefficients from the problems given training data.

The most relevant paper to this work was presented by Pelillo [11] involve learning positive compatibility coefficients. He also imposed these problems as inequality constrained optimization problems and proposed an algorithm to solve them based on Rosen's gradient projection method. However, there were some unreasonable and obsolete assumptions that had been made throughout the paper. In this work, we tries to replace them with more realistic and updated ones. We also explore the use of different objective functions other than the one use in Pelillo's work. Our new optimization problems become nonlinear least square problems which can be solved numerically using the basic Gauss-Newton method [1]. This approach is tested with artificial examples; and the results show that our learned compatibility coefficients capture the relative importance within the ones used for generating the data.

This report is organized as follows. Chapter 2 covers the Hummel's version of Continuous Relaxation Labeling algorithm. In chapter 3, we formulate the problem of learning compatibility coefficients. Then, in chapter 4, we propose a way to optimized the problem. In chapter 5, the experiment and results are presented. Finally, in chapter 6, we summerize the work and discuss related issues.

Chapter 2

Continuous Relaxation Labeling

We stated that Continuous Relaxation Labeling is an algorithm that aims to find an optimal labeling over a given discrete constrained data where the labeling is represented by probabilistic mass functions. Since the past, there have been many versions of Relaxation Labeling presented. In this work, we follow the Hummel's definition of continuous Relaxation Labeling [5], and Parent's radial projection update rule [7]. This chapter explains necessary definitions that will be used to formulate the problem of learning compatibility coefficients.

2.1 Formulation of Continuous Relaxation Labeling and Consistent Labeling

The work spaces of Continuous Relaxation Labeling can be viewed as graphs, where edges imply the compatibilities among the labeling nodes. Each node in these graphs, represents an object to be labeled, is attached with probabilistic mass function, which tell how likely of that node having a particular label. That is, at each node i^{th}, we define the probability of that node having label λ to be $p_i(\lambda)$. Clearly, the only equality constraints of relaxation labeling are that the sum of probability of each node must equal to one, $\sum_\lambda p_i(\lambda) = 1, \quad \forall i \in 1 \dots n$, and the inequality constraints are $p_i(\lambda) \geq 0, \quad \forall i \in 1 \dots n$ because probabilities should not be negative. Thus, for a given graph, the labeling is a vector $\vec{p}$, which contains the probabilistic mass functions of all nodes.

Because of the compatibilities within the given graph, $\vec{p}$ cannot be assigned freely. The compatibility defines how much a node with a specific label receives "supports" from other arbitrarily labeled nodes. Higher support values strengthen the probability of the node for having that label. These compatibilities can be represented by a structure R. In the second-

order compatibility:

$$R = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & R_{ij} & \vdots \\ R_{n1} & \cdots & R_{nn} \end{bmatrix} \tag{2.1}$$

This represents the relationships between two nodes in the graph, where:

$$R_{ij} = \begin{bmatrix} r_{ij}(\lambda_1, \lambda_1) & \cdots & r_{ij}(\lambda_1, \lambda_m) \\ \vdots & r_{ij}(\lambda_a, \lambda_b) & \vdots \\ r_{ij}(\lambda_m, \lambda_1) & \cdots & r_{ij}(\lambda_m, \lambda_m) \end{bmatrix} \tag{2.2}$$

This represents the real compatibility values between two nodes having particular labels in the graph. In this work, we focus on the second-order case for the reason of simplicity; however, the higher order forms can be derived by following the same approach.

Hummel proposed that the relaxation process will achieve consistent labeling if and only if we can locally maximize the average local support $A(\vec{p})$, which is defined as follow:

$$A(\vec{p}) = \sum_{i,\lambda} s_i(\lambda; \vec{p}) p_i(\lambda) \tag{2.3}$$

Where:

$$s_i(\lambda; \vec{p}) = \sum_{i',\lambda'} r_{ii'}(\lambda, \lambda') p_{i'}(\lambda') \tag{2.4}$$

This term $s_i(\lambda)$ defines the "support" for label λ at node i^{th} by the labeling $\vec{p}$, which computes the sum of compatibilities of all neighbors of i^{th}. Thus, the formal definition of Continuous Relaxation Labeling processes can be writen as follows:

$$\max_{\vec{p}} \; A(\vec{p}) \tag{2.5}$$

Subject to

$$\sum_{\lambda} p_i(\lambda) = 1, \quad \forall i \in 1 \ldots n \tag{2.6}$$

$$p_i(\lambda) \geq 0, \quad \forall i \in 1 \ldots n \tag{2.7}$$

2.2 Relaxation and Radial Projection

To optimize the previous problem, a gradient ascend update rule with radial projection was proposed. Given an initial estimate of labeling $\vec{p}$, the gradient

ascend rule tries to locally maximize the average local support, while the radial projection ensures that each step produces a valid labeling, which does not violate the constraints.

According to the gradient ascend rule, at each step t with small $\alpha > 0$:

$$\vec{p}^{t+1} = \vec{p}^{t} + \alpha \nabla A(\vec{p}; R) \tag{2.8}$$

First of all, we need to compute the gradient. Hummel showed that the gradient of the average local support with symmetric compatibility R:

$$\nabla A(\vec{p}; R) = \left[\frac{\partial A(\vec{p}; R)}{\partial p_1(\lambda_1)}, \ldots, \frac{\partial A(\vec{p}; R)}{\partial p_i(\lambda_a)}, \ldots, \frac{\partial A(\vec{p}; R)}{\partial p_n(\lambda_m)} \right] \tag{2.9}$$

$$\frac{\partial A(\vec{p}; R)}{\partial p_i(\lambda_a)} = 2 s_i(\lambda_a; \vec{p}, R) \tag{2.10}$$

The gradient of the average local support is merely the vector of support values corresponding at each node. Even if the compatibility is not symmetry, this is still a valid gradient direction [5].

However, if we keep following these support values, the output labeling may violate the constraints. In order to avoid this, Parent suggested the radial projection rule, which basically maintain the gradients in positive quadrants (2.11) and projects the output vector onto the boundary equality constraints (2.12).

$$s_i^{*t}(\lambda) = \begin{cases} s_i^t(\lambda) - \min_\lambda s_i^t(\lambda) & \text{if } \min_\lambda s_i^t(\lambda) \leq 0; \\ s_i^t(\lambda) & \text{otherwise.} \end{cases} \tag{2.11}$$

$$\vec{p}_i^{t+1} = \frac{\vec{p}_i^{t} + \alpha \vec{s}_i^{*t}}{1 + \alpha(\vec{s}_i^{*t} \cdot \vec{1})} \tag{2.12}$$

Using this rule, the relaxation process will converge to a consistent labeling that satisfies the constraints as long as the step size parameter α is small enough.

Chapter 3

Learning Compatibility Coefficients Problems

In this chapter, we discuss the problems of learning compatibility coefficients in supervise learning aspect. Given a training set of object-label pairs, the problem is to learn good compatibility coefficients such that, after several relaxation processes, the output labeling are close, as much as possible, to the desire ones. This learning mechanism can be applied to the situation where we want to create labeling systems that can adapt to various types of problems. The idea is very similar to many machine learning approaches, where we learn parameters of given models. The most prominent one must be recurrent neural networks [11], since the compatibility in relaxation labeling can be represented by a certain network of neural, and many iterations in relaxation process are equivalent to recurrently filtering the input by this network.

Other types of learning are also possible in learning compatibility coefficients of relaxation labeling. Some unsupervised learning algorithms, such as Expectation Maximization, can also be applied to this scheme because, in any cases, we need a mechanism to determine an initial labeling for the relaxation process, and we can use this initial labeling to guide these unsupervised algorithms. However, in this work, we will not go further in this issue.

3.1 Formulation Learning Compatibility Coefficients

First, we define the set of training examples containing object-label pairs, L:

$$L = \{L_1, \ldots, L_k, \ldots\}$$
$$L_k = \left\{(O_1, \lambda_1^L), \ldots, (O_i, \lambda_i^L), \ldots\right\}$$

For a given initial labeling $\vec{p}^I$, we want to find good compatibility coefficients containing in R such that:

$$\text{Relax}(R, \vec{p}^I) = \vec{p}^F(R) \rightarrow \vec{p}^L \tag{3.1}$$

Which means, if we apply the relaxation labeling process of R to $\vec{p}^I$, the final labeling $\vec{p}^F(R)$ should be very close to the desire labeling $\vec{p}^L$ from the given training data, where:

$$\vec{p}^L = \left[p_1^L(\lambda_1), \ldots, p_n^L(\lambda_m)\right] \tag{3.2}$$

$$p_i^L(\lambda) = \begin{cases} 1 & \text{if } \lambda = \lambda_i^L \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

To formulate the problem mathematically, we have to define ways to measure the distance between two labeling. In general, these kinds of function can be viewed as the measurements of error. Suppose that we have defined E as an error between two labeling, objective function (3.1) is equivalent to:

$$\min_R \; E(\vec{p}^F, \vec{p}^L) = \min_R \; E_{relax}(R; \vec{p}^I, \vec{p}^L) \tag{3.4}$$

This is the general objective of this project. In the next section, we will discuss some possible error functions proposed in [11].

3.2 Error functions

There are two possible error functions proposed by Pelillo. The first one is derived using KL divergence from information theory. Without any support reason, the paper focussed on this error function.

$$E_{relax}(R; \vec{p}^I, \vec{p}^L) = -\sum_{k,i} \ln \vec{p}_i^F(R) \tag{3.5}$$

The another one, mentioned in the paper, is the sum of square of norm of subtraction.

$$E_{relax}(R; \vec{p}^I, \vec{p}^L) = \frac{1}{2} \sum_k ||\vec{p}^F(R) - \vec{p}^L||^2 \tag{3.6}$$

Where we can see that, using this error function, the learning compatibility coefficients tasks become nonlinear least square problems. In this work, we will derive a Gauss-Newton based learning algorithm from this error function.

Chapter 4

The Learning Algorithm

In this chapter, we explain the algorithm for learning compatibility coefficients problem based on the square error function. As we stated that the relaxation labeling processes can be viewed as nonlinear functions of compatibility R and initial labeling $\vec{p}^I$; combining with square of norm error function, this yields nonlinear least square problems.

Though there have been many approaches presented to solve nonlinear least square, most of them are based on the basic Gauss-Newton method [1]. In this work, we present a learning algorithm based on this fundamental one, so that others can perform similar derivations on more advance methods.

4.1 The Gauss-Newton Algorithm

Given a nonlinear least square minimization problem in the form:

$$\min_x \frac{1}{2} \sum_i ||f(x_i) - y_i||^2 \tag{4.1}$$

We want to find an optimum $\hat{x}$ that minimize the function. Usually, when dealing with a nonlinear function, we can find a local optimum corresponding to a given initial point by following the gradient. The Newton's method gives us the optimal step size if the function is locally convex around the initial point. This idea is to approximate the first order Taylor series around the point and move proportionally to the inverse of gradient. Similarly, this idea can be applied to the nonlinear least square problems. Using the first order Taylor series expansion on f:

$$f(\hat{x}) \approx f(x) + J(x)\Delta \tag{4.2}$$

Substituting into (4.1) yields:

$$\min_\Delta \frac{1}{2} \sum_i ||f(x_i) + J(x_i)\Delta - y_i||^2 \tag{4.3}$$

This is a form of linear least square problems, where we know immediately that the optimal Δ is where the derivative of (4.3) is equal to zero.

$$\sum_i J^T(x_i)(f(x_i) + J(x_i)\Delta - y_i) = 0 \tag{4.4}$$

Solving this equation yields:

$$\Delta = \left(\sum_i J^T(x_i)J(x_i)\right)^{-1} \sum_i J^T(x_i)(y_i - f(x_i)) \tag{4.5}$$

The output Δ can be used to approximate $\hat{x}$. However, arbitrary functions are not always locally convex, we need to keep updating the x numerically to reach the best approximation of $\hat{x}$. This yield the Gauss-Newton algorithm, where at each iteration, we solve (4.5) and update x using:

$$x_i^{s+1} = x_i^s + \Delta \tag{4.6}$$

4.2 The Gauss-Newton Algorithm for Learning Compatibility Coefficients

The idea is similar to the previous section, but now we work on the learning compatibility coefficients problem. The objective function is:

$$\min_R \frac{1}{2} \sum_k ||\vec{p}_k^F(R) - \vec{p}_k^L||^2 \tag{4.7}$$

Since the relaxation labeling process, which produces $\vec{p}_k^F$, can be viewed as a nonlinear function. Approximating this function using the first order Taylor series expansion yields:

$$\vec{p}^F(\hat{R}) \approx \vec{p}_k^F(R) + J_{\vec{p}_k^F}(R)\Delta \tag{4.8}$$

Substituting into (4.7) yields:

$$\min_\Delta \frac{1}{2} \sum_k ||\vec{p}_k^F(R) + J_{\vec{p}_k^F}(R)\Delta - \vec{p}_k^L||^2 \tag{4.9}$$

From the previous section, we now know that the optimal $\hat{R}$ can be found by numerically update R using:

$$R^{s+1} = R^s + \Delta \tag{4.10}$$

Where:

$$\Delta = \left(\sum_k J_{\vec{p}_k^F}^T(R) J_{\vec{p}_k^F}(R)\right)^{-1} \sum_k J_{\vec{p}_k^F}^T(R)(\vec{p}_k^L - \vec{p}_k^F(R)) \tag{4.11}$$

4.3 Derivation of Jacobian

The learning algorithm by following the Gauss-Newton update rule is straightforward, but the important part of it is how to compute the Jacobian, where:

$$J_{\vec{p}^F}(R) = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \frac{\partial p_i^F(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{4.12}$$

However, since a relaxation labeling process itself is composed of many iterations, computing partial derivatives require the exact number of iterations, clearly these derivatives are also functions of this number. In practice, we can compute these derivatives while running the relaxation processes simultaneously. Observe the radial projection rule 2.12, this implies the recurrent relations between current and next iterations of relaxtion processes. Using the quotient rule on 2.12 yields:

$$\frac{\partial p_i^{t+1}(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} = \frac{\left(1 + \alpha \vec{s}_i^t \cdot \vec{1}\right)\left(\frac{\partial p_i^t(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} + \alpha \frac{\partial s_i^t(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})}\right) - (\vec{p}_i^t + \alpha \vec{s}_i^t)\left(\alpha \frac{\partial s_i^t(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} \cdot \vec{1}\right)}{\left(1 + \alpha \vec{s}_i^t \cdot \vec{1}\right)^2} \tag{4.13}$$

Also, for the support functions 2.4, their derivatives are:

$$\frac{\partial s_i^t(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} = \sum_{n', \bar{\lambda}} r_{nn'}(\lambda, \bar{\lambda}) \frac{\partial p_{n'}^t(\bar{\lambda})}{\partial r_{jk}(\lambda', \tilde{\lambda})} + p_k^t(\tilde{\lambda}) \tag{4.14}$$

Where:

$$\frac{\partial p_i^0(\lambda)}{\partial r_{jk}(\lambda', \tilde{\lambda})} = 0 \tag{4.15}$$

Because the initial labeling are not functions of compatibility coefficients.

With these rules we can compute the Jacobian, and thus complete the learning algorithm. However, there are two issues that need to be mentioned here. First, in the recursive derivation of Jacobian, we ignore the positive vector maintaining rule (2.11), since the min operation on discrete space "seems to" make the relaxation functions become noncontinuous (in fact, they are still continuous). To deal with this problem, we can use the finite difference approximation of gradient instead of computing the exact values. We can also leave this problem if the relaxation processes contain only a few iterations because, at a positive start point, it is most likely that the output vectors at the end of iterations are still positive. Second, the size of Jacobian is very large, since it is composed of all partial derivatives of all labels with respect to all compatibility coefficients. In this case, we may want to find a fast way to do inversion.

Chapter 5

Experiment and Results

In this section, we show the testing of our learning algorithm with two examples. The experiment was set so that the five iteration relaxation labeling processes were run on a set of ten example spaces of size 100×100 with only two labels, and each node interacted with other nodes within 3×3 neighbors. Every trial was given with a random initial labeling which was normalized, and used for generating training set and learning period. Each learning process started with an initial point that was randomly generated around the solution.

Example 1

The first example, we generated the test set with a symmetric compatibility.

$$R^*(\lambda_1, \lambda_1) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad R^*(\lambda_1, \lambda_2) = \begin{bmatrix} 0 & 5 & 0 \\ 5 & 0 & 5 \\ 0 & 5 & 0 \end{bmatrix}$$

$$R^*(\lambda_2, \lambda_1) = \begin{bmatrix} 0 & 5 & 0 \\ 5 & 0 & 5 \\ 0 & 5 & 0 \end{bmatrix} \qquad R^*(\lambda_2, \lambda_2) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Iteration	Error
1	128.7932
2	127.3024
3	127.2940
4	142.7021
5	57.7645
6	1.4456
7	0.0015
8	6.0127e-008

Table 5.1: The optimization results for example 1.

The learned compatibility coefficients were:

$$R^*(\lambda_1,\lambda_1) = 10^9 \begin{bmatrix} -0.2140 & -0.2493 & -0.2357 \\ -0.2524 & -4.1774 & -0.2303 \\ -0.2327 & -0.2333 & -0.2330 \end{bmatrix}$$

$$R^*(\lambda_1,\lambda_2) = 10^9 \begin{bmatrix} 0.0190 & -1.1812 & -0.0028 \\ -1.1842 & 0.0166 & -1.1623 \\ 0.0003 & -1.1653 & 0.0000 \end{bmatrix}$$

$$R^*(\lambda_2,\lambda_1) = 10^9 \begin{bmatrix} 0.0216 & -1.1844 & -0.0020 \\ -1.1869 & 0.0201 & -1.1629 \\ 0.0004 & -1.1654 & 0.0000 \end{bmatrix}$$

$$R^*(\lambda_2,\lambda_2) = 10^9 \begin{bmatrix} -0.2115 & -0.2525 & -0.2351 \\ -0.2550 & -4.1741 & -0.2310 \\ -0.2326 & -0.2335 & -0.2330 \end{bmatrix}$$

We can see that the structure of original compatibility was captured in the result. However, notice that the amplitude of learned coefficients was very high; this issue will be discussed in the next chapter.

Example 2

The second example, we generated the test set with a non-symmetric compatibility.

$$R^*(\lambda_1,\lambda_1)=\begin{bmatrix}1&1&1\\4&4&4\\1&1&1\end{bmatrix} \qquad R^*(\lambda_1,\lambda_2)=\begin{bmatrix}0&5&0\\5&0&5\\0&5&0\end{bmatrix}$$

$$R^*(\lambda_2,\lambda_1)=\begin{bmatrix}0&5&0\\1&2&1\\0&5&0\end{bmatrix} \qquad R^*(\lambda_2,\lambda_2)=\begin{bmatrix}4&4&4\\1&8&1\\4&4&4\end{bmatrix}$$

Iteration	Error
1	313.1885
2	306.1973
3	306.1188
4	531.1023
5	295.5549
6	0.5450
7	0.0017
8	1.1938e-004

Table 5.2: The optimization results for example 2.

The learned compatibility coefficients were:

$$R^*(\lambda_1,\lambda_1)=10^8\begin{bmatrix}-0.0958&-0.1149&-0.0794\\-0.3877&-1.3331&-0.4061\\-0.0965&-0.0987&-0.0948\end{bmatrix}$$

$$R^*(\lambda_1,\lambda_2)=10^7\begin{bmatrix}0.0092&-5.0160&0.1732\\-4.8442&0.2038&-5.0277\\0.0011&-4.8541&0.0191\end{bmatrix}$$

$$R^*(\lambda_2,\lambda_1)=10^7\begin{bmatrix}0.0124&-5.1085&0.2619\\-0.9807&-1.6240&-1.2626\\0.0014&-4.8670&0.0319\end{bmatrix}$$

$$R^*(\lambda_2,\lambda_2)=10^8\begin{bmatrix}-0.3854&-0.4142&-0.3605\\-0.0981&-1.7093&-0.1263\\-0.3866&-0.3900&-0.3835\end{bmatrix}$$

Again, we can see that the result was formed relatively to the original compatibility.

Chapter 6

Summary and Discussion

In this work, we have defined learning compatibility coefficient problems for continuous relaxation labeling processes and presented an algorithm to solve them. Given training data set of object-label pairs, the learning problems can be viewed as supervise learning tasks, where we want to find proper compatibility coefficients that can produce the desire labeling corresponding to input examples. Since the relaxation labeling processes are nonlinear functions and the error functions that we choose are the sums of square norm, the learning problems become nonlinear least square, which can be solved using fundamental methods, such as the Gauss-Newton algorithm. We also state how to derive partial derivatives from the iterative-based nonlinear function of relaxation labeling processes. Besides, we conduct experiments to test our algorithm using two artificial examples. The results show that the algorithm produces meaningful coefficients which can be used to determine the co-relationships between objects.

The major contribution for this work are from [11], which basically focuses on the same topic, but we have modified some unreasonable assumptions and replaced obsolete theories to make idea becomes more practical. Despite these developments, there are still many issues that are open for discussion.

First, as we showed that the results of this algorithm is quite large when compared to the original solutions. This is because the output labeling of relaxation labeling processes that use radial projection equation (2.12) will not be affected by the size of large compatibilities since it is cancelled out in the fraction. This is an ill pose problem, and the optimization process could not decide a proper size for learned compatibilities. To handle this problem, we have to enforce constraints that will limit the size of output compatiblities during the optimizations.

Second, finding an initial point for relaxation processes and the learning algorithm is not a trivial issue. Usually, to solve these similar issues, people use randomly re-selected and optimize or Monte Carlo based sampling ap-

proaches. However, because the relaxation labeling processes involve large number of parameters over large spaces of labeling objects, this large scale property makes those approaches impractical. We suggest that the coarse-to-fine based approaches may be useful here by starting with only major compatibility relations within data and increasingly adding more relations each time we finished optimization.

Third, as we said that these are large scale optimization problems. Computing the Jacobians is a tedious task, and the products of computed Jacobians should not be directly inverted according to their dimensions. To avoid these problems, we can use the finite difference approximation rules to compute the Jacobians, and approximate the inverses using the recurrence based approaches such as presented in secant algorithm instead. Also, if the compatibilities are limited within local space, parallel algorithms can be used to enhance the performance.

Finally, as we stated at the begining that relaxation labeling processes are very close to recurrent neural networks. Thus, the successful optimization approaches for the networks may be applied in learning compatiblity algorithm to avoid local optimums. Not only that, the derivations of the learning algorithms for general order relaxation labeling processes are also interesting to be concerned because we have senses that higher order compatibilities are equivalent to the second order compatiblities with equality constraints.

At last, even though this report ends here, we have reached a certain point in developing an algorithm for learning compatibility coefficients for continuous relaxation labeling process. Still, there are many possible extensions available; this means that the issue has yet to be complete. Thus, we encourage people to continue working on this topic, and will be happy if they can develop better solutions for the learning compatibility coefficients problems.

Bibliography

[1] Gauss-newton algorithm, wikipedia. Website. http://en.wikipedia.org/wiki/Gauss%E2%80%93Newton_algorithm.

[2] S. Geman, D. Geman, K. Abend, TJ Harley, and LN Kanal. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*. *Journal of Applied Statistics*, 20(5):25–62, 1993.

[3] R.M. Haralick and L.G. Shapiro. The consistent labeling problem: Part I.

[4] JJ Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554, 1982.

[5] RA Hummel and SW Zucker. On the foundations of relaxation labeling processes. In *Readings in computer vision: issues, problems, principles, and paradigms*, page 605. Morgan Kaufmann Publishers Inc., 1987.

[6] JL Mohammed and R. Hummel. Gradient projection algorithm for relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):330–332, 1983.

[7] P. Parent and SW Zucker. Radial projection: An efficient update rule for relaxation labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):886–889, 1989.

[8] P. Parent and S.W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):823–839, 1989.

[9] S. Peleg and A. Rosenfeld. Determining compatibility coefficients for curve enhancement relaxation processes. *IEEE Trans. Syst. Man Cybern*, 8(7):548–555, 1978.

[10] M. Pelillo. The dynamics of nonlinear relaxation labeling processes. *Journal of Mathematical Imaging and Vision*, 7(4):309–323, 1997.

[11] M. Pelillo and M. Refice. Learning Compatibility Coefficients for Relaxation Labeling Processes (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9).

[12] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, 1976.