Supervised Hypergraph Labeling

Toufiq Parag
Janelia Farm Research Campus, HHMI
19700 Helix Drive, Ashburn VA 20147
paragt@janelia.hhmi.org

Ahmed Elgammal
Dept of Computer Science
Rutgers University, Piscataway, NJ 08854
elgammal@cs.rutgers.edu

Abstract

We address the problem of labeling individual datapoints given some knowledge about (small) subsets or groups of them. The knowledge we have for a group is the likelihood value for each group member to satisfy a certain model. This problem is equivalent to hypergraph labeling problem where each datapoint corresponds to a node and the each subset correspond to a hyperedge with likelihood value as its weight. We propose a novel method to model the label dependence using an Undirected Graphical Model and reduce the problem of hypergraph labeling into an inference problem. This paper describes the structure and necessary components of such model and proposes useful cost functions. We discuss the behavior of proposed algorithm with different forms of the cost functions, identify suitable algorithms for inference and analyze required properties when it is theoretically guaranteed to have exact solution. Examples of several real world problems are shown as applications of the proposed method.

1. Introduction

The objective of this paper is to label datapoints into two classes when the information about the datapoints is available in terms of small groups[1] of data. The information encode the likelihood that the members of the corresponding group follow a certain model. We assume the existence of such model, either analytical or implicit, for the entity or pattern we are trying infer. Given the subsets and their corresponding likelihood measures or weights, we wish to label all the individual datapoints that satisfy the model into one class and rest of them to the other class. We call these weights 'supervised' since they were calculated given a model, not based on similarities in some feature space.

It should be noted here that we are interested in scenarios where higher order knowledge about data subsets of sizes $k > 2$ is the primary source of information for labeling; we may or may not have any information about the individual data sample or pairs of samples. Furthermore, we may have data subsets of different sizes, i.e., $k = 3, 4, \dots$, each with corresponding likelihood measure, to decide the individual label from.

[1] Group simply implies a collection of datapoints. The mathematical definition of group is not used in this paper. Similarly, likelihood simply implies a numerical weight.

There is a direct connection between the problem we are dealing with and hypergraph node labeling. Hypergraphs are generalization of graphs where each hyperedge connects more than two vertices, i.e., a hyperedge is a subset of of nodes. Weighted hypergraphs, a hypergraph whose hyperedges are associated with real valued weights, have been recently gained much popularity in computer vision for the purpose of representing geometrical information. Each small group of data, as discussed above, can be considered as a hyperedge of some hypergraph. The likelihood measure for this group of data is analogous to the weight of corresponding hyperedge. Given such a hypergraph, we would like to label its nodes into two categories such that optimal number of nodes, connected by hyperedge with large weights, tend to fall into the same category.

There are many examples of such problem in machine learning and vision literature. In part-based object recognition for computer vision, it is useful to learn an implicit statistical model among groups of detected parts to decide which of the detected parts actually belong to the object we are trying to recognize. It is well known that larger groups of parts capture more geometrical information than pairs of parts. This is a necessary step in recognition due to the fact that object part-detectors often generate many false alarms.

Subset-wise information is also utilized in model estimation algorithms. The problem of model estimation is to determine the parameters of a model given a set of (noisy) data and an analytic form of a model. Standard algorithms for this sort of problem, e.g., RANSAC [7] and its variants, randomly sample smaller subsets of data, compute target model parameters and calculate an error value for each of the subsets. These methods are used for fundamental matrix computation, affine motion estimation etc in computer vision [23] literature. In this paper, we show that any model estimation problem can be treated as an instance of the framework we discuss and our algorithm can solve any model estimation problem that RANSAC (and its variants) can handle. In addition, the proposed method can also solve the problems, where we only have a model for small parts instead of the whole object/entity, that RANSAC can not handle.

We propose a novel approach to solve this labeling problem in a principled fashion. Each datapoint is associated with a binary label variable. The label variables (and their dependencies) are modeled by an Undirected Graphical Model or Markov Network with appropriate neighborhood system and clique definitions (to be explained later). To compute the optimal configuration of the label variables, it is necessary to minimize an overall cost function which is a summation of all cost functions defined on the cliques.

We propose a class of potential function for each clique which plays the central role in deciding labels of data samples. From a theoretical point of view, we investigate what makes the proposed potential function submodular and therefore is guaranteed to produce an accurate solution for two-class problems [5, 21, 14]. For a specific form of this potential function, the inference problem can be solved with a simple algorithm that runs in linear time w.r.t. the number of data samples and subsets. Standard methods for inference, e.g., linear programming relaxation or sum product algorithms, can be used for other forms of potential functions. This Markov Network formulation is able to cope with the situation when the likelihood values are available for different size subsets.

The main contribution of this paper is to cast the hypergraph labeling problem as an undirected graphical model inference problem. This reduction demonstrates the connection between hypergraph problems and graphical models and widens the scope of application of the latter. We propose necessary cost functions for the purpose, discuss how the behavior of the labeling algorithms is influenced by their different forms and which of these forms have a theoretical guarantee to have a solution. This work enables us to use the rich literature of graphical model inference to be readily applicable in problems like hypergraph labeling.

2. Related work and motivation

The types of problem that proposed method addresses are different from what hypergraph clustering methods are designed for (not described here due to space limitation, see e.g., [1, 10, 24, 19, 12]). In hypergraph clustering, each hyperedge is usually associated with a weight computed from the similarity of the member datapoints in some feature space. On the other hand, in our framework, the hyperedges are associated with a likelihood value, calculated given a model. Conceptually, the most significant difference lies in the objective function the proposed method and that the clustering method optimizes. For example, spectral graph clustering methods [1, 10, 24] minimizes a product of (hyper) cut function and a restraint function. In contrast, our method minimizes a summation of weighted combinations of competing penalty functions. In what follows next, we give an example where hypergraph clustering can not solve the labeling problem given a hypergraph with likelihood values as weights.

Let us suppose we have points in 2D as displayed in Figure 1 and we wish to determine which 2D points belong to the lines.

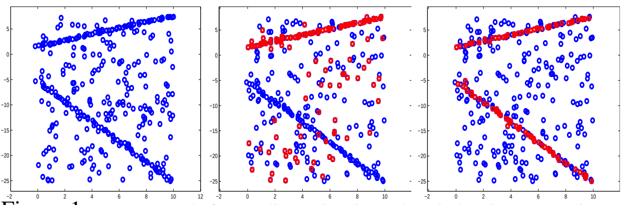

Figure 1. A toy example for two line estimation. Blue circles: input 2D point, red squares: detected line points. Left: input, middle: hypergraph clustering result, right: proposed algorithm. ()

We follow a standard procedure for line fitting: first, k datapoints ($k >= 2$) are sampled randomly, then a line is estimated through these points and the percentage (also called inlier ratio) of all points approximately falling on the line is computed. This inlier ratio is used as hyperedge weights. We wish to verify if hypergraph clustering method is able to separate the line points from noises given these weights.

For this example, we apply the method of [24] as a representative method for hypergraph clustering. The work of [24] partitions the set of vertices into two subsets S and S^c by minimizing a product of a cut function and a restraint function, $\text{Ncut}(S, S^c) = \text{cut}(S, S^c)\left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}\right)$, where the volume $\text{vol}(S)$ of S quantifies the associativity among the vertices within the subset S. Without the restraint function, the solution is always trivial, all vertices are assigned to same cluster. Most of the spectral graph and hypergraph clustering algorithms use this (or very similar) objective function. While the cut function $\text{cut}(S, S^c)$ tries to minimize the strength of the connection between two clusters, the restraint function $\left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(S^c)}\right)$ controls the size or associativity within the resulting clusters. As a result, we often observe these algorithms to produce somewhat 'balanced' clusters. That is, it will find subsets S, S^c so that not only the sum of edge weights crossing the partition is minimized, but also the connectivity among the subset members is maximized.

This effect is clearly visible in the result of [24] shown in Figure 1 (middle). The clustering algorithm separates the points into two clusters, each having one line and many noisy samples. This perfectly conforms with the discussion above, the inter-cluster connectivity between the two clusters is minimum and each of the clusters has high internal associativity among the line points. None of the two lines can be estimated from this output generated by hypergraph clustering. We ideally want the line points to be separated from the noise, which is the outout of the proposed algorithms as explained below, as shown in Figure 1 (right). There are other relevant works that we will discuss throughout the paper where they naturally connect to the context.

3. Graphical model framework

Suppose there are n datapoints $\{v_i|\ 1 \leq i \leq n\}$ that we wish to separate into two subsets, subset A comprising the samples that satisfy the model and B comprising the rest. We wish to label these samples using binary variables $\{x_i \in \{0,1\}|\ 1 \leq i \leq n\}$ where $x_i = 1$ implies $v_i \in A$ and $x_i = 0$ implies $v_i \in B$. Throughout the paper, we use the term 'group' to imply (interchangeably) a subset of $\{v_{i_1}, \ldots, v_{i_k}\}$ of size k where $1 \leq i_l \leq n$ and $1 \leq l \leq k$. The likelihood (also called weights) that all members of $\{v_{i_1}, \ldots, v_{i_k}\}$ are generated from a model M is denoted by $\lambda(v_{i_1}, \ldots, v_{i_k})$. The $\lambda(v_{i_1}, \ldots, v_{i_k})$ values are assumed to be normalized in $[0,1]$.

As already stated, the objective of this work is to propose an algorithm to label the datapoints $v_i,\ 1 \leq i \leq n$ mainly based on the information provided in terms of small groups of them. We may or may not have some information about individual sample v_i or pair of samples, but the focus is on how to utilize the knowledge we have for the subsets. To this end, the input to the algorithm is a set of small groups or subsets of data samples along with their likelihood values. Let $\mathcal{V}^k$ be the set of all groups $\{v_{i_1}, \ldots, v_{i_k}\}$ that satisfy a certain condition[2].

To establish a neighborhood $\mathcal{N}_i^k$ for any datapoint v_i, we define v_j is a neighbor of v_i if both v_i and v_j are members of any group $V^k \in \mathcal{V}^k$.

$$\mathcal{N}_i^k \;=\; \{\, v_j \mid j \neq i \text{ and } \exists V^k\ \{v_i, v_j\} \subseteq V^k\}. \quad (1)$$

For inputs with different size groups, $k_1, k_2, \ldots,$ we can similarly define a neighborhood system for each subset size.

Now, we can define a Markov Network over the label variables x_i assuming the Markov property that the value of x_i depends on x_j only if v_j is a neighbor of v_i [9]. Any subset $V^k \in \mathcal{V}^k$ defines a clique of size k in the neighborhood system. Also, let X^k denote the labels of members of V^k, i.e., $X^k = \{x_{i_1} \ldots x_{i_k}\}$ and X denote labels of all n datapoints $\{x_1, \ldots, x_n\}$.

It is well known that the probability of any assignment $p(X)$ depends on what is known as the Gibbs energy function $E(X)$ [17]. The energy function $E(X)$ is the summation of potential functions defined on cliques. Let $E^k(X^k)$ denote an appropriately defined clique potential function for the clique $V^k = \{v_{i_i}, \ldots, v_{i_k}\}$ of size k. With different size cliques, the Gibbs energy function equals to sum over all energy function of different sizes [17].

$$E(X) = \sum_{k=1}^{K} \sum_{V^k \in \mathcal{V}^k} E^k(X^k) \quad (2)$$

The optimal assignment $X = \{x_1, \ldots, x_n\}$ should minimize this Gibbs energy function. To complete the definition of the overall energy function, we have to define the clique potential functions $E^k(X^k)$ for each k.

At this point, it is clear that the graphical model we are utilizing here is more commonly known as Markov Random Fields (MRF). We choose to use the terms Graphical models or Markov network primarily to distinguish this work from more traditional applications of MRFs in vision. Typically, MRF studies in computer vision assumes a spatial neighborhood among the neighboring pixels or their labels [16, 20]. One popular example of this type of neighborhood is the grid structure assumed among the pixels. We use a hypothetical neighborhood definition induced by the subsets and is different from those assumed in traditional applications of MRF. We use synonymous term for MRF, e.g., probabilistic graphical models, for our work so that one does get confused with other traditional MRF structure used in vision.

4. Proposed clique function

The clique potential functions determines the cost of assigning the labels within the clique into different classes. It can be viewed as playing a similar role of the cut function in clustering algorithms. But, as we do not want to use an auxiliary function, e.g., the restraint function as in Section 2, using a generalized cut function similar to the one defined in [24] will only produce trivial results.

Instead, we introduce two penalty functions $g_c(\cdot),\ c \in \{1,0\}$, for the two categories A and B. The penalty function value increases with the number of member variables assuming the opposite class label. Let η_c denote the number of variables $x_{i_l} \in X^k$ in the clique to be assigned to class c. A penalty function $g_c(\cdot)$ is defined on the number $\eta_{(1-c)}$ of variables to be assigned to the opposite class $1-c$. Ideally, we want the penalty functions $g_c(\cdot)$ to be nondecreasing with the increase in η_{1-c}. We define a clique potential function $E^k(\cdot)$ for clique V^k as a linear combination of competing penalty functions $g_1(\eta_0)$ and $g_0(\eta_1)$ weighted by the likelihood value $\lambda(V^k)$ and $1 - \lambda(V^k)$ respectively.

$$E^k(X^k) = \ \beta_1\, \lambda(V^k)\, g_1(\eta_0) \ + \ \beta_0\, (1 - \lambda(V^k))\, g_0(\eta_1). \quad (3)$$

In this definition, β_1 and β_0 are two nonnegative balancing parameters. Recall, $\lambda(V^k)$ quantifies the likelihood measures that all $v_{i_l} \in V^k$ belong to A, i.e., satisfy a model, and $\eta_1 + \eta_0 = k$. With high likelihood $\lambda(V^k)$ value, the

[2] For example, a condition check retains only groups with weights larger (or errors less) than a threshold δ^k.

clique potential would be prone to decrease η_0, the number of variables $x_{i_l} \in X^k$ to assume label 0 and tolerate a small penalty $(1 - \lambda(V^k))\ g_0(\eta_1)$ as $(1 - \lambda(V^k))$ is small. The behavior will reverse for low $\lambda(V^k)$ when it will try to decrease η_1. Thus, this potential function becomes a weighted combination of two *competing* penalty functions, each trying to increase number of $x_{i_l} \in X^k$ to be labeled to its corresponding class.

In cases where there are two models available for both categories A and B, the resulting likelihood values $\lambda_1(V^k)$ for A and $\lambda_0(V^k)$ for B will replace $\lambda(V^k)$ and $1 - \lambda(V^k)$ respectively in Equation 3. Example of such likelihood values could be found in object recognition where the probabilities of a subset of parts to belong to the object and that to belong to the background is useful. We show one such experiment in Section 7.3.

Studies with higher order MRF models generally consider pointwise potential functions as the 'driving force'or the main contributing factor for inference. Higher order clique functions in MRF literature act as smoothness terms designed to ensure consistency over the neighboring locations or to remove isolated regions. These functions takes the minimum value when all the labels in the clique are equal, without any bias to any particular class, and increases with the label disagreement among the variables. Examples functions are truncated linear combination of labels in the clique [14] or minimum of several truncated linear functions [13]. Without pointwise energies, these higher order potentials will lead to a trivial solution and assigns all the labels to one of the classes. This is exactly why these type of non-discriminative potential functions are not useful in the present scenario where we usually do not have pointwise information (see [2] for more discussions and examples).

5. Submodularity and tractability

Submodularity [2] is an important property for solving inference in undirected graphical models. It has been shown in the literature that for a submodular clique potential function, there are algorithms to produce a globally optimal solution in polynomial time [5, 14] for two class problems. This section analyzes the proposed clique function to identify what properties make it submodular and therefore can be solved efficiently and accurately with existing algorithms.

As stated before, we want $g_c(\cdot)$, $c \in \{0, 1\}$, functions to be monotonically increasing. Let us express the proposed clique function of Equation 3 as a set function. Recall that X^k is an indicator vector for set $A \in V^k$ (Section 3). We will write $E^k(X^K)$ as $f(A) = C_1 g_1(\eta_0) + C_0 g_0(\eta_1)$ where $C_1 = \beta_1 \lambda(V^k)$, $C_0 = \beta_0(1 - \lambda(V^k))$ and $\eta_0 + \eta_1 = k$. If we add any $v_{i_l} \in V^k \setminus A$ to A, the clique function becomes $f(A \cup \{v_{i_l}\}) = C_1 g_1(\eta_0 - 1) + C_0 g_0(\eta_1 + 1)$. If we augment A further by $v_{i_j} \in V^k \setminus (A \cup \{v_{i_l}\})$, the clique function becomes $f(A \cup \{v_{i_l}, v_{i_j}\}) = C_1 g_1(\eta_0 - 2) + C_0 g_0(\eta_1 + 2)$. To prove submodularity of $f(A)$, we have to prove that the successive increase in $f(A)$ diminishes.

$$
\begin{aligned}
& f(A \cup \{v_{i_l}\}) - f(A) \geq f(A \cup \{v_{i_l}, v_{i_j}\}) - f(A \cup \{v_{i_l}\}) \\
& \Rightarrow -C_0\ g_0\ (\eta_1 + 2) + 2\ C_0\ g_0(\eta_1 + 1) - C_0\ g_0(\eta_1) \\
& \geq C_1\ g_1(\eta_0) - 2\ C_1\ g_1(\eta_0 - 1) + C_1\ g_1(\eta_0 - 2) \\
& \Rightarrow C_0 \left\{ \Big[g_0\ (\eta_1 + 2) - g_0(\eta_1 + 1) \Big] - \Big[g_0(\eta_1 + 1) - g_0(\eta_1) \Big] \right\} \\
+ C_1 & \left\{ \Big[g_1(\eta_0) - g_1(\eta_0 - 1) \Big] - \Big[g_1(\eta_0 - 1) - g_1(\eta_0 - 2) \Big] \right\} \ \leq 0
\end{aligned}
$$

Denoting the second order difference as $\Delta_c(n) = \big[g_c(n) - g_c(n-1)\big] - \big[g_c(n-1) - g_c(n-2)\big]$ where $c \in \{0, 1\}$, we observe that the condition for submodularity of the proposed clique function is as follows.

$$
C_1\ \Delta_1(\eta_0)\ +\ C_0\ \Delta_0(\eta_1 + 2) \leq 0. \tag{4}
$$

There are several options for $g_c(\cdot)$ functions to render the clique potential to be submodular.

Linear $g_c(\cdot)$**:** It is obvious that for linear $g_c(\cdot)$, the second order differences is 0 and the clique function is submodular. In fact, for linear $g_c(\cdot)$, the clique potential becomes modular.

Concave $g_c(\cdot)$**:** As the second derivative of any concave function is negative, the condition in Equation 4 holds.

Constant Δ_c**:** For $g_c(\cdot)$ functions with constant $\Delta_c(n) = \bar{\Delta}_c$, one can design a submodular clique function with a concave $g_1(\cdot)$ and convex (or linear) $g_0(\cdot)$ whenever $C_1 > C_0$ and $|\bar{\Delta}_1| > |\bar{\Delta}_0|$ where $|\cdot|$ implies absolute value (and vice versa). We can also use a combination only when $C_1 > C_0$ and use concave functions for both $g_1()$ and $g_0()$ otherwise.

The focus of this paper is neither to propose a clique reduction technique nor a new MRF inference algorithm. Therefore, it is unnecessary to analyze whether or not the proposed clique potentials can be transformed into some other form in order to apply a specific inference technique.

6. Inference and Learning

The inference algorithm to be used to minimize the energy function $E(X)$ depends on the choice of $g_c(\cdot)$ functions. The following two subsections describes two algorithms that can solve the inference problem. We also discuss how to learn the form of penalty function $g_c(\cdot)$ function for the cases when labeled examples are available (i.e., x_i are given).

6.1. Special case: Linear $g_c(\cdot)$

The linear form of $g_c(\cdot),\ c \in \{0, 1\}$ increases from l_c to h_c in proportion to $k - \eta_c$.

$$
g_c(k - \eta_c) = l_c + \frac{h_c - l_c}{k}(k - \eta_c). \tag{5}
$$

It is straightforward to see (and well known, see [2])that the value of x_i is simply the label for which the summation of likelihood ratios weighted by the slope of $g_c(\cdot)$ is minimum.

$$x_i^* = \arg\min_{c\in\{0,1\}} \sum_{V^k\in\mathcal{V}^k \wedge v_i \in V^k} \beta_c \lambda_c(V^k)\, \frac{h_c - l_c}{k}. \quad (6)$$

For simplicity, we used $\lambda_1(V^k) = \lambda(V^k)$ and $\lambda_0(V^k) = (1-\lambda(V^k))$. This solution can be computed in $O(n+|\mathcal{V}^k|)$, with one pass over all the subset weights and another pass over all the datapoints. For multiple k, the corresponding weights for each tuple size will be added.

6.2. General Case: Nonlinear $g_c(\cdot)$

For nonlinear $g_c(\cdot),\ c \in \{0,1\}$, e.g., concave or convex forms, the likelihood weights can not be evenly distributed to each of the datapoints as in Equation 6. Problems with nonlinear $g_c(\cdot)$ can be solved using a linear programming (LP) formulation suggested in [5, 14]. See [11, 22] and references therein to learn more about various inference algorithms available in the literature. We adopted the LP relaxation technique and used CPlex solver to produce the labeling. It is not clear yet how a general submodular higher order clique function can be reduced efficiently to pairwise interactions (see [8, 21]) to apply Graph-cut algorithm, but other algorithms like sum-product belief propagation [15] or more efficient variants of it can also be used for the proposed method.

6.3. Behavior of clique function

Intuitively, an increase of $\min g_c()$ or a decrease of $\max g_c()$ will lower the number of datapoints to be labeled as category c since the former would introduce a penalty to assign label c to any sample in a subset and the latter would simply reduce the penalty to assign $1-c$. Also, a concave $g_c()$ would be more conservative on the number of x_{i_l} to be labeled as $1-c$, that is, it will be more difficult to have label disagreement within the clique. A convex penalty function would impose less amount of cost to have few x_{i_l} to take the opposite label.

6.4. Learning penalty functions

Modeling the framework as a Markov Network also allows us to learn the penalty functions and the parameters when several labeled sequences are available. Since, in our framework, penalty functions $g_c()$ are defined for the integer values, we only want the values of $g_c(\eta_{(1-c)})$ for all $c \in \{0,1\}$ and $0 \le \eta_{(1-c)} \le k$. A redefinition of penalty functions makes it possible for us to use training algorithms (e.g., [4]) for log-linear models to learn these values.

7. Experiments and Results

7.1. Model Estimation

One important application of the proposed method is model estimation. Let us suppose that we have n datapoints $v_i, i = 1,\ldots,n$, in some feature space. Part of these data samples are generated from some model that we wish to estimate. We sample T^k subsets of size k, fit the candidate model to these subsets and compute the model estimation errors. The value of k is kept larger than or equal to s which is the minimum number of points required to fit a model (e.g. $s = 2$ to fit a line). The subsets producing an error less than a problem specific threshold δ^k constitute the set $\mathcal{V}^k$. The estimation error ϵ of any member $V^k \in \mathcal{V}^k$ is transformed to a likelihood measure by a suitable transformation, e.g., $\lambda(V^k) = 1-\epsilon$ if we know $0 \le \epsilon \le 1$ or $\lambda(V^k) = \exp(-\epsilon/\sigma)$.

It is important to mention here that, in model estimation, only higher order information is available. We do not have any knowledge about how likely each sample is to follow a certain model. For linear penalty functions for the proposed clique potentials, the inference algorithm (Equation 6) breaks the higher order costs into pointwise costs. Reducing the higher order costs into pointwise ones is not equivalent of using unary potentials only, because we do not have a tool to compute this unary potentials. Nonlinear penalties may not be as attractive as linear ones, since it will require sophisticated inference algorithms. But, as we will see, clique functions with nonlinear penalties offer certain advantages that could be useful in special cases.

Fundamental matrix computation: Given two images of the same scene from different viewpoint, the objective is to find the point matches that conform with the camera model expressed by the fundamental matrix. Four pairs of images from the Corridor, Valbonne, Merton II, Library datasets of the standard Oxford database[3] were used in this experiment. We selected the two images with the largest variation in viewpoint, usually the first and last images (see Table 1).

For the proposed method each match is considered as a datapoint. We sampled 2500 subsets of size $k = 8$ (we know $s = 7$ in this case). Other parameter values are, threshold $\delta^k = 1$, and linear parameters $[h_1, l_1, h_0, l_0] = [1.01, 0.02, 1.0, 0]$ for the clique function. For RANSAC, we sampled a subset of size 7 for at most 2500 times and used $\delta_{ransac} = 0.001$ as distance threshold (that produces the best result). We also compare the performance with three other variants of RANSAC, namely MLESAC, MAPSAC and LO-RANSAC, that were shown to be able to compute higher number of inliers in the survey paper [3]. The maximum number of iterations were kept the same as that of RANSAC and parameters in original implementations were retained.

[3]http://www.robots.ox.ac.uk/ vgg/data/data-mview.html

method	Corridor: $\{000, 010\}$, 50/150		Valbonne: $\{000, 010\}$, 30/90		Merton: $\{002, 003\}$, 50/150		Library: $\{002, 003\}$, 50/150	
	missed	FP	missed	FP	missed	FP	missed	FP
RANSAC	0.07 ± 0.06	0.10 ± 0.03	0.13 ± 0.06	0.13 ± 0.02	0.01 ± 0.02	0.38 ± 0.01	0.02 ± 0.01	0.25 ± 0.01
MLESAC	0.01 ± 0.03	0.07 ± 0.02	0.13 ± 0.04	0.11 ± 0.03	0.00 ± 0.00	0.33 ± 0.00	0.01 ± 0.01	0.17 ± 0.01
MAPSAC	0.04 ± 0.04	0.05 ± 0.02	0.19 ± 0.03	0.09 ± 0.01	0.00 ± 0.00	0.30 ± 0.00	0.02 ± 0.01	0.15 ± 0.01
LO-RANSAC	0.17 ± 0.07	0.11 ± 0.04	0.19 ± 0.03	0.10 ± 0.03	0.03 ± 0.03	0.37 ± 0.04	0.01 ± 0.01	0.22 ± 0.01
Proposed	0.09 ± 0.02	0.06 ± 0.01	0.16 ± 0.02	0.05 ± 0.01	0.14 ± 0.02	0.25 ± 0.01	0.05 ± 0.01	0.13 ± 0.01

Table 1. Performance comparison for Model Estimation. Each column head shows the name and indices of the image used and the ratio of correct over incorrect matches. Each row shows the mean and std deviation of the missed inliers and false positives (FP) produced by the corresponding method.

Table 1 shows the image names (ratio of true inliers) and the result statistics such as fraction of missed inliers (ratio of missed example over inliers only) and ratio of false positive inliers (ratio of false positives over all matches) in 100 runs for all the methods. As we can see, in almost all the cases, the proposed labeling method generates lower false positive rates than that of all the variants of RANSAC with similar or close miss rates. In model estimation, lower false positive rates are desirable up to a small increase of miss rate since there will be less impurities to influence the computation of the model. One possible reason is, RANSAC and variants, decide based on the performance of the most recently sampled V^k and replace the set of inliers, if needed, and discard the estimation results from all previous subsets. The proposed method aggregates the results of all past model estimation results in order to obtain a better decision about which datapoints in the set should be labeled inliers.

In Figure 2, we show qualitative comparison in worst case. The results of proposed methods are compared with MAPSAC, because of its superior performance w.r.t. other variants, where they produced the largest number of missed correct matches and false detections (miss+FP). This figure shows that, in the worst case, the proposed method would miss less correct matches and generate fewer false alarms than those of MAPSAC. In our experiments, we experienced same pattern for all the images.

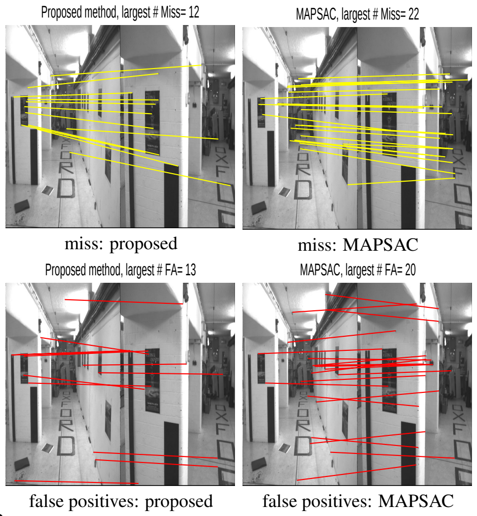
Proposed method, largest # Miss= 12
MAPSAC, largest # Miss= 22
miss: proposed
miss: MAPSAC
Proposed method, largest # FA= 13
MAPSAC, largest # FA= 20
false positives: proposed
false positives: MAPSAC

Figure 2. Worst case qualitative performances for detecting valid correspondences between two images. Showing the largest miss+FP case. Yellow (red) line: matching point pair missed (falsely detected) by the method.

To show the advantage of using a nonlinear penalty term, let us imagine a scenario where we wish to tolerate a large miss rate (but not too high to miss required number of samples) but attain as low false positive rate as possible. To achieve this result, we experimented with a concave $g_0()$ and a linear $g_1()$ on the same set of data described above. The subsets and their weights for the proposed method were generated in the exact same manner. As explained before, concave $g_c()$ functions are reluctant to assign the datapoints within a clique to label into different classes. Hence, a concave $g_0()$ will produce more missed inliers and less false positives. We compare the result with a two step recursive RANSAC method with decreasing threshold. The initial threshold was taken much lower than the previous experiments to produce as low false positives as possible.

Table 2 shows the miss and false positive results of recursive RANSAC (along with required threshold at first step), proposed method with linear and nonlinear penalties. It is clear that, very different thresholds for different datasets for recursive RANSAC are required to achieve similar miss and false positive rate as that of the proposed method with linear and non-linear penalties. We used the same penalty function to produce outputs for all four datasets; this implies the robustness of our method over algorithms in RANSAC family for this purpose. This also shows that proposed method can achieve much lower false positive rates with nonlinear penalties than it linear counterpart. More results for model estimation are presented in the supplementary material.

7.2. Circle/Ellipse detection

This section describes an experiment where we used a part based model to detect the full object. We detect circles/ellipses in in images and synthetic 2D points (with 40% inlier ratio) as a collection of small curves. From the edge points detected on an image, k = 4 *neighboring* ones are selected randomly. Then we fit a second degree polynomial on these K points. If the estimated curve turned out to be a line, we assign a very low weight on it. In Figure 3, the edge points labeled as $x_i = 1$ by the inference algorithm with linear penalty terms are colored in red (white in the last image) . It is interesting to see that the proposed method can detect all the circles (multilpe ones) in the images.

It is also possible to identify the *circles* in these images using a model for the whole circle (as opposed to those for curves) utilizing a circle fitting algorithm [18] for model fitting procedure. The proposed method produced the same result as in Figure 3 when we used such model. When applied to the same dataset with the same circle fitting algorithm, RANSAC was able to extract only the largest circle.

method	Corridor: $\{000, 010\}$, 50/150 thd, missed, FP	Valbonne: $\{000, 010\}$, 30/90 thd, missed, FP	Merton: $\{002, 003\}$, 50/150 thd, missed, FP	Library: $\{002, 003\}$, 50/150 thd, missed, FP
Rec-RANSAC	$1^{-3}, 0.61 \pm 0.14, 0.02 \pm 0.02$	$1^{-4}, 0.63 \pm 0.17,\ 0.06 \pm 0.03$	$5^{-5}, 0.62 \pm 0.10, 0.08 \pm 0.04$	$1^{-5}, 0.70 \pm 0.19, 0.05 \pm 0.13$
Proposed-lin	$-, 0.46 \pm 0.05, 0.03 \pm 0.00$	$-, 0.67 \pm 0.07, 0.02 \pm 0.01$	$-, 0.57 \pm 0.05, 0.20 \pm 0.02$	$-, 0.36 \pm 0.03, 0.08 \pm 0.02$
Proposed-nonlin	$-, 0.57 \pm 0.04, 0.02 \pm 0.00$	$-, 0.63 \pm 0.07, 0.02 \pm 0.01$	$-, 0.61 \pm 0.04, 0.07 \pm 0.01$	$-, 0.46 \pm 0.05, 0.05 \pm 0.01$

Table 2. Performance in minimizing false positives. Each column shows the name and indices of the image used, the ratio of correct over incorrect matches and in the second line the threshold used for recursive RANSAC, miss and FP rates. Each row shows the mean and std deviation of the missed inliers and false positives (FP) produced by the corresponding method.*Note that, RANSAC requires threshold to vary between a wide range to achieve similar result of the proposed method with fixed parameters.*

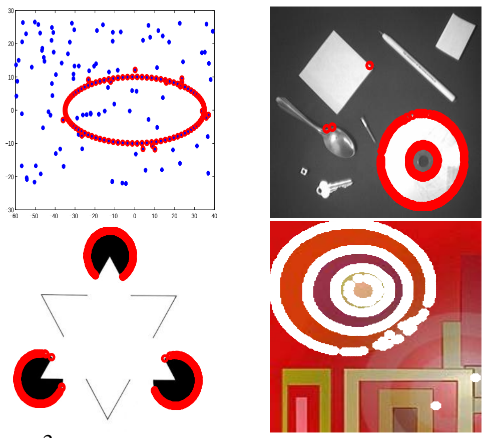

Figure 3. Circles detected by proposed method overlaid on the image.

By iterating the process, each time removing the inliers of the previous step, it is possible to recover all the circles. But, RANSAC needs to know the number of instances of a model (in this case circle) present in the dataset to obtain all of them. On the contrary, the proposed algorithm was able to identify all the model instances in a single iteration. It is possible, at least for the geometric structures, to compute the number of model instances simply by running a connected component search on the positively labeled data.

7.3. Object localization

Part based representation of an object has gained much attention recent years in computer vision. In a recent work by Ferrari et.al. [6], subsets of edge contour segments were used to describe different objects. This work generated all the edges from an input image and then computes all k adjacent segments (kAS) that were shown to represent the object well for detection. We apply the proposed method to identify all the edges that should belong to the object given the set of all kAS. That is, all the kAS are considered to be the subset of k datapoints (for this case, each datapoint is an edge) and given many of these subsets we wish to find which edges belong to the object (the set A).

The experiments were conducted on three objects of the the ETH shape dataset, namely applelogos, bottles and swan. For each type of object, half the images for each category were selected as training images. Out of all 3AS ($k = 3$) descriptor [6] in the training set, we select representative ones to be the centers of a hierarchical clustering output. These representative descriptors, called code, are generated for both foreground and background. During testing, we generate all the 3AS from the test image. For each 3AS, we generate the likelihood value for the edge segments to belong to the object(background) to be the distance from the closest object(background) code. As described in Section 4, there are two (implicit) models in this experiment, one characterizes the object and the other correspond to the background. Finally, we run the hypergraph labeling algorithm to get the edges of an object. We employed both linear and nonlinear penalty functions in this experiment too. For nonlinear $g_c()$ functions we used concave functions for both the classes.

Figure 4 shows some output of proposed method for identifying object edges. To clarify that only kAS likelihoods , without the labeling technique, are not sufficient to identify the object edges, we show all the edges that belong to kAS segments that have $\lambda_1(V^k) > \lambda_0(V^k)$. The proposed labeling algorithm with nonlinear penalties more accurately identifies the object edges from the background than that with linear penalties. As explained before, concave penalty functions tend to prevent breaking up the clique members into different classes. As a result, the proposed method with nonlinear penalties attempts to attain more edges on the object and reduce the number of background edges to be labeled as 1 than its counterpart with linear penalties. Table 3 summarizes average number of edges detected by these three methods on the object and outside of the groundtruth bounding box for the whole dataset. In comparison, proposed method with nonlinear penalties is able to detect more object edges with fewer false detections than other two techniques describe here.

method	Bottles		Applelogos		Swan	
	in	out	in	out	in	out
kAS Pr	9.16	7.2	5.9	8.72	6	9
linear	8.83	6.86	5.81	7.09	5.06	4.85
nonlin	8.16	6.66	5.77	6.95	4.81	4

Table 3. Number of edges detected by three methods, kAS likelihood only, proposed-linear and proposed-nonlinear. For each object class, the columns show average number of detected edges within and outside of the groundtruth bounding box.

8. Conclusion

This paper describes a novel method for datapoint labeling given a hypergraph of data with supervised weights. We discuss how the problem can not be handled by clustering methods and provide a graphical model framework to solve it. The necessary cost functions for this framework is proposed and suitable inference algorithms for labeling is discussed. In addition to theoretical treatment of the form of the cost function to discover when it will lead to a global solution, we also show several applications where the pro-

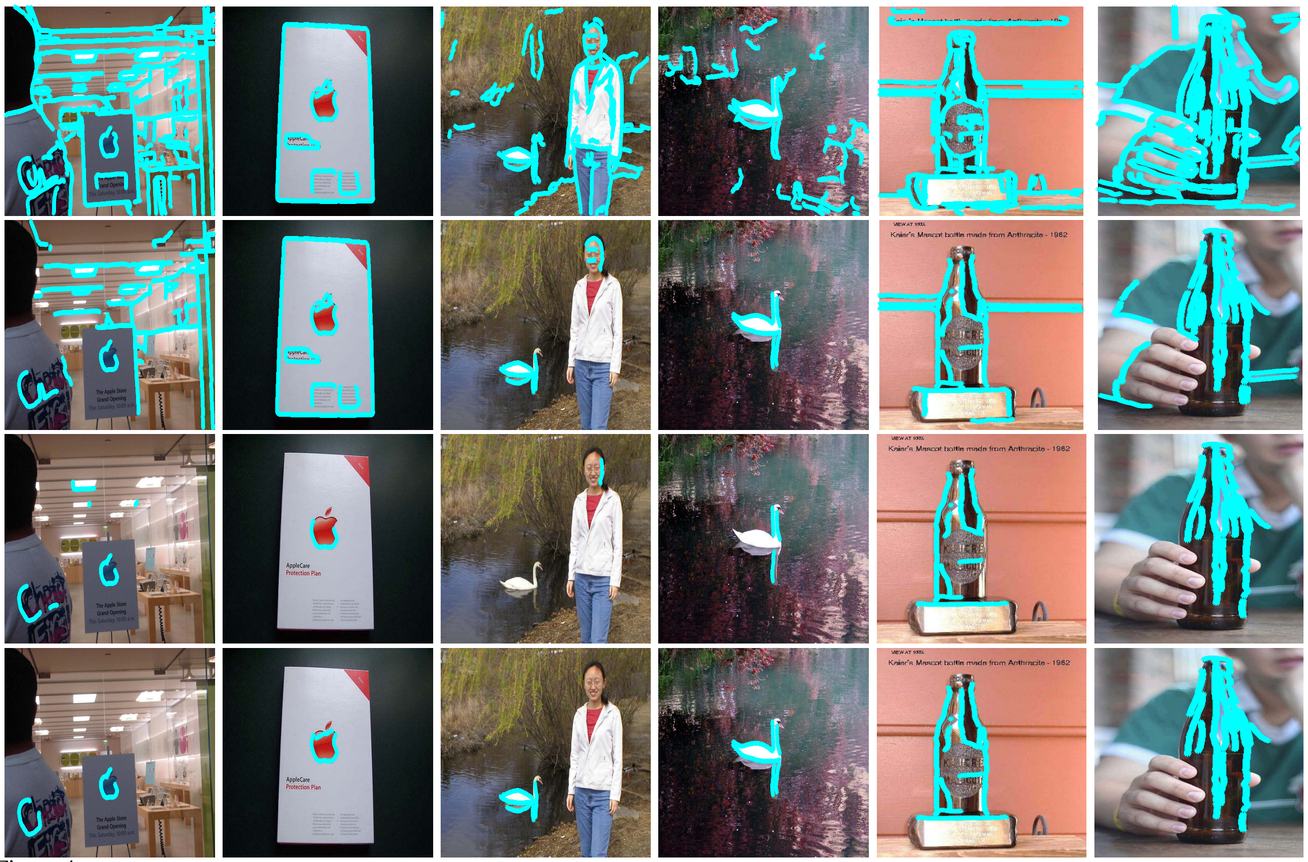

Figure 4. Qualitative performances of proposed method for detecting edges on objects (from left to right) applelogos, swan and bottle. Rows: top to bottom, input edges, edges with $p(\text{kAS} \in \text{object}) > p(\text{kAS} \in \text{bckgnd})$, proposed linear, proposed nonliner.

posed labeling algorithm can either produce better results or interesting results to be investigated further.

References

[1] S. Aggarwal, L. Jongwoo, L. Zelnik-Manor, P. Perona, and S. Kreigman, D.and Belongie. Beyond pairwise clustering. In *CVPR*, 2005.

[2] Authors. Supplementary material.

[3] S. Choi, L. Kim, and W. Yu. Performance evaluation of ransac family. In *BMVC*, 2009.

[4] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822, 2008.

[5] M. C. Cooper. Minimization of locally defined submodular functions by optimal soft arc consistency. *Constraints*, 13:437–458, 2008.

[6] V. Ferrai, F. Fevrier, L.and Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1):36–51, 2008.

[7] M. A. Fischler and B. R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *ACM*, 24:381–395, 1981.

[8] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *CVPR*, 2005.

[9] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution and bayesian restoration of images. *PAMI*, 6(6):721–741, 1984.

[10] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *CVPR*, 2005.

[11] H. Ishikawa. Higher order clique reduction in binary graph cut. In *CVPR*, 2009.

[12] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 11, 2000.

[13] P. Kohli, M. Kumar, and P. Torr. P^3 and beyond: solving energies with higher order cliques. In *CVPR*, 2007.

[14] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order mrfs. In *CVPR*, 2009.

[15] F. R. Kschischang, B. J. Frey, and H. andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 1998.

[16] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.

[17] S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 1991.

[18] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21:145–152, 1987.

[19] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In *ECCV*, 2006.

[20] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *PAMI*, 30:1068–1080, 2008.

[21] S. Živný and P. G. Jeavons. Which submodular functions are expressible using binary submodular functions? Research Report RR-08-08, Oxford University Computing Lab, UK, June 2008.

[22] T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 29(7):1165–1179, 2007.

[23] Workshop. 25 years of ransac. in conjunction with CVPR 2006.

[24] J. H. Zhou, D. and B. Schlkopf. Beyond pairwise classification and clustering using hypergraphs. Technical Report 143, Max Planck Institute for Biological Cybernetics, 2005.