Semi-Supervised Affinity Propagation with Instance-Level Constraints

Inmar E. Givoni, Brendan J. Frey
Probabilistic and Statistical Inference Group
University of Toronto
10 King's College Road, Toronto, Ontario, Canada, M5S 3G4

Abstract

Recently, affinity propagation (AP) was introduced as an unsupervised learning algorithm for exemplar based clustering. Here we extend the AP model to account for semi-supervised clustering. AP, which is formulated as inference in a factor-graph, can be naturally extended to account for 'instance-level' constraints: pairs of data points that cannot belong to the same cluster (cannot-link), or must belong to the same cluster (must-link). We present a semi-supervised AP algorithm (SSAP) that can use instance-level constraints to guide the clustering. We demonstrate the applicability of SSAP to interactive image segmentation by using SSAP to cluster superpixels while taking into account user instructions regarding which superpixels belong to the same object. We demonstrate SSAP can achieve better performance compared to other semi-supervised methods.

1 Introduction

Affinity propagation (AP) (Frey & Dueck, 2007) is an exemplar-based clustering method that takes as input similarities between data points. It outputs a set of data points that best represent the data (*exemplars*), and assignments of each non-exemplar point to its most appropriate exemplar, thereby partitioning the data-set into clusters. The objective of AP is to maximize the sum of similarities between the data points and their exemplars. The AP algorithm is based on casting this NP-hard optimization problem in terms of a factor-graph, and performing approximate MAP inference using the max-product algorithm(Kschischang et al., 2001). The factor-graph used in AP can be modified to allow constraints and additional information to be accounted for in a principled way, like introducing flexible priors on cluster size (Tarlow et al., 2008). In this work we are interested in extending the AP framework to semi-supervised clustering.

Semi-supervised clustering algorithms are concerned with finding good partitions of data in the presence of side information. Two popular forms of side information are partial labels and instance-level constraints. We consider the case where side information is given in the form of instance-level constraints on pairs of data points. *Cannot-link* constraints indicate the two data points cannot be in the same cluster while *must-link* constraints indicate the data points must be in the same cluster (Wagstaff & Cardie, 2000). There is a distinct difference between side information given in the form of partial labels and that given in the form of instance-level constraints. The two are not equivalent since labeled data can always be used to construct instance-level constraints while the converse does not hold in general, making instance-level constraints weaker in terms of the amount of information they carry. However, instance-level constraints are often faster or cheaper to obtain than labels, and can sometimes be automatically collected (Wagstaff et al., 2001; Klein et al., 2002; Shental et al., 2003).

Another difference between partial labels and instance-level constraints is that instance-level constraints do not directly provide information about the total number of clusters or classes in the data. It is always possible to construct the transitive closure for any set of instance-level constraints: if points i and j are constrained to be in the same cluster, and points j and k are likewise constrained, then it follows that i,j, and k must all be in the same cluster. Similarly, if i and j must be in the same cluster, but i and k cannot be in the same cluster then j and k cannot be in the same cluster. However, such grouping of constraints does not indicate the total number of groups is equal

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

to the total number of clusters in the data, as it is only a lower bound. Thus, many algorithms that incorporates instance-level constraints, such as clustering algorithms that attempt to partition the data subject to the constraints, can detect clusters composed of points for which no side information is given (Wagstaff et al., 2001; Klein et al., 2002; Shental et al., 2003). When, on the other hand, side information is provided as partial labels, it is often assumed that the unique number of class labels should be the number of identified classes (Xiao et al., 2007; Leone et al., 2008). In order for such algorithms to succeed, there must be at least one labeled example from each class in the data. The property of being able to detect clusters for which no side information is given is desirable for cases where some clusters are 'easy' to detect, while others may require human intervention. The motivating example in this work is the case of user-guided image-segmentation, where segmentation results can be sometimes improved with quite limited user input, and where it should not be necessary to explicitly provide side-information for every object in the image.

One prominent approach for devising semi-supervised algorithms for instance-level constraints is to modify standard unsupervised clustering algorithms so that they explicitly account for the constraints. This was done for k-means clustering (Wagstaff et al., 2001), and Mixture of Gaussians (Shental et al., 2003). A missing aspect of this approach is that it does not explicitly propagate constraints. Intuitively, if we look at the set of points that are very close to either point in a must-link constraint, it is likely these points should also be in the same cluster as the must-link constraint points. This property is sometimes referred to as space-level constraints (Klein et al., 2002), and is often imposed by adapting the similarities (or distances) between data points to better align with the constraints; a good distance metric for the data should intuitively assign a small distance to a pair of points with a must-link constraint, and a large distance to a pair of data points with a cannot-link constraint. Then, the adjusted metric can be used as input to an unsupervised clustering algorithm (Xing et al., 2003). Naturally, the approach of adapting similarities can be combined with the approach of explicitly accounting for constraints (Klein et al., 2002; Basu et al., 2004).

The solution we propose here is to adapt the underlying factor-graph used in affinity propagation such that constraints are explicitly added, but also propagated. We add 'meta-points' to the underlying model and appropriate function nodes to govern the allowed set of solutions. The meta-points link together the must-link pairs and prevent cannot-link pairs from being in the same cluster. Similarities of points to meta-points are constructed in a way that allows the model to also account for space-level constraints: points that are very close to points that must be in the same cluster are likely to also be in the same cluster.

Although, to the best of our knowledge, there are no other AP-based methods that incorporate instance-level constraints, Xiao *et al.* describe an adaption of AP to include partial labels by contracting all similarly labeled points to a new point and adjusting similarities between unlabeled points to the new contracted points. The set of potential exemplars is restricted to that of the contracted points, and standard AP is then run on the modified data-set. This restriction prevents differently labeled points from being put in the same cluster and forces the number of clusters to be equal to the number of unique labels. Leone *et al.* also contract all labeled points and adjust the similarities, but they do not restrict only contracted labeled points to be exemplars. However, they do not attempt to cluster the contracted labeled points, which only serve as potential exemplars. Both algorithms do not account for must-link or cannot-link constraints. In addition, the goal of AP is not only to partition the data but to also find the most representative data points, or exemplars. In applications where the actual exemplars are meaningful the contracted points may not be useful since they do not represent real data points.

2 Semi-Supervised AP

2.1 Unsupervised AP

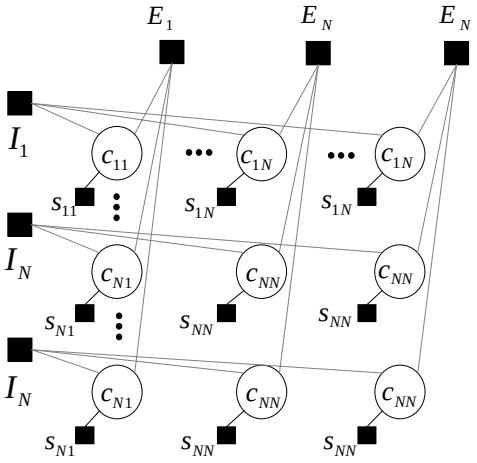

Figure 1: A binary variable model for AP

We begin with a brief review of the unsupervised affinity propagation (AP) model for clustering data points, where we use the binary grid factor-graph described in (Givoni & Frey, 2009) (Fig. 1). The input to the algorithm is pairwise similarities $s(i, j)$ between N data points $i \in (1 \dots N), j \in (1 \dots N)$, where we assume similarities are negative, and the maximal possible similarity between two points is 0. We define N^2 hidden binary variables c_{ij}. Setting $c_{ij} = 1$ denotes that i's exemplar is j, and $c_{ii} = 1$ indicates i is its own

exemplar. The I function nodes introduce the '1-of-N' constraint: each point can be assigned to at most one exemplar (that exemplar can be the point itself, meaning the point is choosing itself as an exemplar). The E function nodes introduce the 'exemplar consistency' constraint: in order for any point $i, i \neq j$ to choose j as its exemplar, j must be its own exemplar. Finally, the S_{ij} function nodes incorporate the user-defined input similarities $s(i,j)$ between data points and their potential exemplars and evaluate to the similarity $s(i,j)$ when $c_{ij} = 1$.

Formally, the function definitions are:

$$I_i(c_{i1}, \ldots, c_{iN}) = \begin{cases} -\infty & \text{if } \sum_j c_{ij} \neq 1, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

$$E_j(c_{1j}, \ldots, c_{Nj}) = \begin{cases} -\infty & \text{if } c_{jj} = 0 \text{ and } \sum_i c_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$$S_{ij}(c_{ij}) = \begin{cases} s(i,j) & \text{if } c_{ij} = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

The graphical model in Fig. 1 together with (1)-(3) result in the following objective function[1]:

$$\begin{aligned} \mathcal{S}(c_{11}, \ldots, c_{NN}) &= \sum_{i,j} S_{ij}(c_{ij}) + \sum_i I_i(c_{i1}, \ldots, c_{iN}) \\ &+ \sum_j E_j(c_{1j}, \ldots, c_{Nj}), \end{aligned}$$

stating that we wish to find the configuration of the c_{ij} variables that maximizes the similarity of the data points to their exemplars. The tendency of a particular point to be an exemplar is given by the preferences $s(i,i)$. Like the similarities, it is assumed to be at most 0. A small negative preference indicates a point is most suitable to be an exemplar and a big negative preference indicates the opposite. Usually, all data points share the same preference p, and this number is a free parameter that controls the amount of clusters found by the algorithm. An intuitive interpretation for the preference is to view it as a cost associated with creating a cluster (with an associated exemplar). Given cluster set-up costs, and similarities between points, the cheaper it is to create clusters, the more clusters the algorithm can find. Therefore the term $\sum_{i,j} S_{ij}(c_{ij})$ in the objective function can be broken into a that represents the sum of similarities of data points to their cluster exemplar and a term representing the total cost associated with setting the clusters. The algorithm attempts to find the best trade-off for a particular setting of the cost.

The approximate MAP setting for the c_{ij} variables is inferred by the max-sum algorithm(Kschischang et al., 2001). It is shown in (Givoni & Frey, 2009) that the different types of messages that need to be propagated in the Fig. 1 graph can be reduced to two simple sets of messages that are iteratively updated until convergence. These messages are the ones exchanged between the hidden variables and the column function nodes E, where the other messages are subsumed into them for simplicity.

The AP update messages are as follows:

$$a(i,j) = \begin{cases} \sum_{k \neq j} \max[0, r(k,j)] & i = j \\ \min\Big[0, r(j,j) + \sum_{k \neq j,i} \max[0, r(k,j)]\Big] & i \neq j \end{cases} \quad (4)$$

$$r(i,j) = s(i,j) - \max_{k \neq j}(s(i,k) + a(i,k)) \quad (5)$$

The messages have an intuitive interpretation; The 'responsibilities' r are indicators of how much data points think other data points are suited to be their exemplars. The 'availabilities' a indicate to what extent data points consider themselves fit to serve as exemplars for other data points. After convergence, the exemplars are found by calculating the set of positive $a(i,i) + r(i,i)$ messages for each $i \in \{1 \ldots N\}$. Non-exemplars are assigned their respective exemplars by choosing $\max_{j \in \mathcal{J}}(a(i,j)+r(i,j))$, where $\mathcal{J}$ denotes the set of exemplars.

2.2 From Unsupervised to Semi-supervised AP

Now, suppose we obtain instance-level constraints for some input data points, and we wish to endow our model with the ability to use this side information. The first intuitive approach might be to directly connect the hidden variables corresponding to data points that must be in the same clusters via a function that enforces this constraint, and similarly, connecting the hidden variables corresponding to cannot-link data points with an appropriate function node as well. It turns out, however, that running AP on such a graphical model yields solutions that satisfy the constraints but are otherwise meaningless; the transitive closure of data points with must-link constraints get grouped together in their own clusters, while non-constrained data points that are similar to the must-link constrained ones are not necessarily in the same cluster with the must-link points, unlike what common sense would dictate. The reason for this is direct constraints do not induce propagation of information from constrained points to non-constrained ones.

Another intuitive idea for incorporating constraints is to alter the similarities between data points. For example, by making the similarity between must-link

[1] The formulation used here is the log-domain max-product, or max-sum algorithm

points be maximal, the similarity between cannot-link points minimal and the similarity between any other two points the shortest-path between them. This way, if i is similar to j, s is similar to t, and j and s must be in the same cluster, the similarity between i and j can be increased if $s(i,j) + s(t,s) > s(i,s)$. However, as noted in (Klein et al., 2002) this approach captures the property that must-link points and their neighbors should be in the same cluster but it does not enforce the desired property that data points similar to cannot-link points are more likely to be put in different clusters. Indeed, it is also noted in (Klein et al., 2002) that enforcing cannot-link constraints is conceptually harder than must-link constraints; even determining if the set of cannot-link constraints has a satisfying assignment is NP-complete. A similar observation is made in (Shental et al., 2003) where the must-link constraints are incorporated by constructing their transitive closure, and constraining them to be in the same cluster, using a relatively simple modification of the EM algorithm update equations, while the cannot-link constraints require the inclusion of a hidden MRF over the data points, resulting in a considerably more involved inference procedure, and an increased degree of approximation.

The solution we propose here is to augment the data points with fictitious 'meta-points' or MTPs. We compute the transitive closure of the must link constraints, and add one MTP for each resulting group as well as to each point in a cannot-link constraint if it is not also part of a must-link group. The MTPs allow us to explicitly enforce the must-link constraints and cannot-link constraints, as well as to propagate must-link constraints and construct a mechanism for cannot-link constraints to be propagated. As expected from the discussion above, the effectiveness of propagating cannot-link constraints is more limited but it is incorporated and inferred using the same simple formulation as the rest of the model, and is shown to yield good results in practice. We now describe how to augment the model with the MTPs.

Let M be the number of MTPs, and let P_m be the set of data points associated with MTP_m, $m \in \{1 \ldots, M\}$. We define symmetric similarities between MTP_m and the input data points as:

$$S(i, MTP_m) \quad = \quad \begin{cases} 0 & \text{if } i \in P_m, \\ \max_{j \in P_m} s(i,j) & \text{otherwise.} \end{cases}$$

Note that $S(MTP_m, i) = S(i, MTP_m)$.

The intuition behind the construction of the meta-points is that data points will now be able to choose either real exemplars or one of the MTPs, if it is more suitable than a real exemplar. The MTPs in turn will have to choose a real exemplar. Since all points in a must-link group will necessarily choose the MTP associated with them, this will also result in a clustering that respects must-link constraints. Other data points are also free to choose MTPs and so points that are similar to a group of must-link points are likely to also choose that group's MTP as an exemplar, leading to the space-level propagation of constraints. Furthermore, if there is a cannot-link constraint between some $i \in P_m$ and $a \in P_n$ we introduce a direct inequality constraints between MTP_m and MTP_n by including function nodes that prevent the MTPs from choosing the same exemplar. This is a compact representation of cannot-link constraints: let $L_m = \{i, j, k\}$ be one set of points that must be in the same cluster, and $L_n = \{a, b, c\}$ be another set of points that must be in the same cluster, if we also know that i cannot be with a, then all pairs in the cross product $L_m \times L_n$ also have a cannot-link constraint ,whether it was explicitly specified or not. However, one constraint between the MTPs of each set is all we need in order to enforce all these constraints, as opposed to the representation in (Shental et al., 2003) that involved MRF connections among all such constraints. Furthermore, any point that will choose some MTP_m as an exemplar, most likely because it is similar to one of the points in the must-link group associated with MTP_m, will be in a different cluster than any point that has a cannot-link constraint with the must-link group associated with MTP_m, if such exists. This can allow space-level propagation of cannot-link constraints.

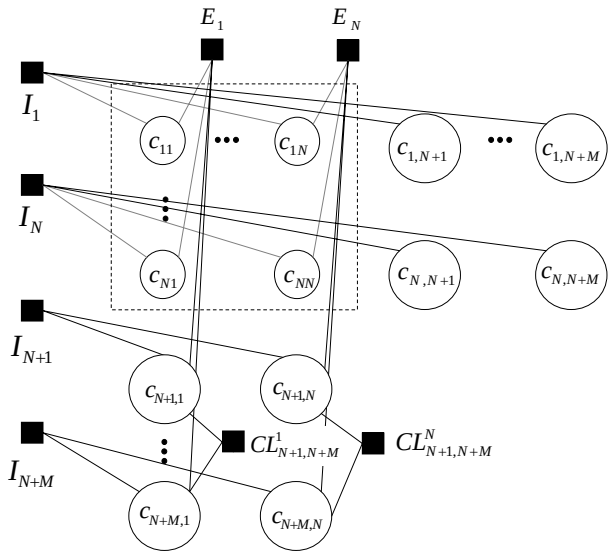

Figure 2: Semi-supervised affinity propagation. The factor-graph inside the dotted line is the original AP model, and the similarity functions s have been removed for clarity

Fig. 2 shows the graphical model of SSAP that includes the MTPs (the similarity function nodes have been omitted for clarity) and the cannot-link constraints . The I function nodes, that enforce the property that each point must choose exactly one exemplar, remain the same as in the standard AP model. For the in-

put data points, the choice of an exemplar is over all data points and all MTPs. Since MTPs should only be allowed to choose exemplars from the set of input data points, the domain of their associated I functions is only over the input data points. The E function nodes, that enforce the constraint that a point can only choose an exemplar if that exemplar chooses itself as an exemplar, is not required for the MTPs, as in fact, by definition they should *not* choose themselves as exemplars but instead pick an exemplar from the set of input data points. The CL function nodes are added to enforce the cannot-link constraints between pairs of MTPs for which such constraints are given, as described above. In Fig. 2 we show for clarity only two MTPs and a cannot-link constraint between them, but in the general case there is not necessarily a constraint between every two MTPs and the exact nature of cannot-link connections between MTPs depend on the given constraints.

Formally the cannot-link function nodes are given by

$$CL^k_{N+m,N+n}(c_{N+m,k}, c_{N+n,k}) = \begin{cases} -\infty & c_{N+m,k} = c_{N+n,k} \\ 0 & \text{otherwise} \end{cases}$$

The message updates for the new model are similar to the original AP messages. In fact, the $a(i,j)$ messages (4) remain the same, save for the indexing domain over k in the sum that changes from $k \in \{1,\ldots,N\}$ to $k \in \{1,\ldots,N+M\}$. The $r(i,j)$ messages (5) remain as before for $i < N$ with a similar indexing change over the maximization. In order to express $r(m,j)$ for $m > N$, we need to include the messages arriving from and sent to the CL constraint nodes. In order to keep the notation simplified, let us define the following two messages:

$$q^j(m,mn) = \mu_{C_{N+m,j} \to CL^j_{N+m,N+n}} \quad (6)$$

$$q^j(mn,m) = \mu_{CL^j_{N+m,N+n} \to C_{N+m,j}}, \quad (7)$$

The update rule for (6), the message from a variable node to the CL function node, is:

$$q^j(m,mn) = a(m,j) + r(m,j) - q^j(mn,m) \quad (8)$$

And the update rule for the message from the CL function node (7) has the form

$$q^j(mn,m) = -\max[0, q^j(n,mn)] \quad (9)$$

Now we can express $r(m,j)$ for $m > N$:

$$r(m,j) = s(m,j) + \sum_{n \in \mathcal{CL}_m} q^j(mn,m) - \max_{k \neq j}(s(m,k) + \sum_{n \in \mathcal{CL}_m} q^k(mn,m) + a(m,k)),$$

Where $\mathcal{CL}_m$ denotes the set of all cannot-link constraints associated with MTP_m. Note that if we substitute in the expression $\hat{s}(m,*) = s(m,*) + \sum_{n \in \mathcal{CL}_m} q^*(mn,m)$ we recover the message update rule of the standard r message (5):

$$r(m,j) = \hat{s}(m,j) - \max_{k \neq j}(\hat{s}(m,k) + a(m,k))$$

Therefore, the influence of messages coming from all the cannot-link constraints can be seen as a modification of similarities to account for these constraints.

The message scheduling we have chosen calculates iteratively q, a, and r messages until convergence. Once the algorithm terminates we assign to all the points which chose an MTP as their exemplar the exemplar chosen by that MTP.

3 Experimental evaluation

3.1 User Interactive Image Segmentation

The particular application we consider here is user interactive image segmentation. There exist many algorithms for unsupervised image segmentation, but in many cases they fail to provide a segmentation that is close to what a human would consider appropriate. One reason behind their shortcomings can be failing to group together different parts of an object, if the parts are very different under any reasonable similarity measure between the elements on which the segmentation is carried out.[2]. For example, a human might consider an image of a person wearing multi-colored and multi-textured clothes as one object but an automatic segmentation will most likely put them in different segments since color, texture, and edge cues will all indicate they should be different objects. Another source of error can be putting together objects that should be separated. For example, if two very similar animals are present in the same image, so that their bodies partially overlap, many segmentation algorithms will group them into one segment. Although it is yet an open question how to overcome these errors in a completely unsupervised manner, often only a small amount of user intervention is needed in order to correct these types of errors.

We are interested in evaluating the usefulness of semi-supervised AP for the task of user interactive image segmentation. Although the term interactive image segmentation usually refers to algorithms geared towards fine separation of background from foreground given a user marked contour of the object or an area known to contain the foreground object, *e.g.* (Rother

[2]These elements can be the image pixels or superpixels - small pixel groups of coherent image regions, obtained by some method of over-segmentation.

et al., 2004), here we are interested in correctly grouping superpixels, segments of over-segmented images, into possibly several objects.

In order to obtain quantitative results for a range of experimental settings, we simulate user interaction by randomly selecting a subset of superpixels for which the algorithm is given the instance-level constraints. This is repeated 10 times to obtain results for different subsets of training data for each image. We test the performance across a range of percentage of points for which constraints are provided to the algorithm.

Our image data consists of 23 images. We first obtained ~ 200 superpixels for each image (Mori, 2005). Each superpixel was hand-labeled, with each image containing between 2 to 11 distinct labels. We then constructed a similarity measure between superpixels that has a 2D distance component, and an equally weighted color component, similar to (Xiao et al., 2007). The first component is calculated as the negative squared Euclidean distance between the centers of mass of every pair of superpixels, normalized by the sum of all such distances. The color component is the negative squared Euclidean distance between average superpixel color in Cielab space, similarly normalized.

3.2 Evaluation Criterion

We report the modified Rand index (Rand, 1971; Wagstaff & Cardie, 2000) achieved by each method. The Rand index calculates the agreement between two clustering solutions $C, \hat{C}$, where usually one is a clustering algorithm solution ($\hat{C}$), and the other is the true class labels (C). The index is in the range $[0, 1]$ where 1 indicates a perfect agreement between the clusterings. It can also be interpreted as the probability the two clustering solutions agree on whether two randomly drawn points belong to the same cluster or to different clusters. For every pair of points clustered, the points are either in the same cluster in both solutions, not in the same cluster in both solutions, or the pair of points can be in the same cluster according to one solution and not in the same cluster according to the other solution. The first and second cases above represent the agreement events between the clustering solutions. The total sum of these events is normalized by the total number of events (the number of all pairs of points, $\frac{N(N-1)}{2}$) and the result is the Rand index. Following (Wagstaff & Cardie, 2000) we calculate this quantity only for pairs for which no supervised information was given, either directly or by transduction. Furthermore, as observed by (Xing et al., 2003), this measure tends to give inflated scores when there are many clusters, since there are many more pairs of points that are not in the same cluster than there are pairs of points that are in the same cluster, and most algorithms will correctly predict that most pairs are not in the same cluster. This can be remedied by giving the same weight to the points that are in the same cluster (according to $\hat{C}$) and those that are not in the same cluster. The probabilistic interpretation is the chance of two data points to have agreeing clustering solutions, where the data points are drawn uniformly at random from the same cluster (according to $\hat{C}$) with chance 0.5 and from different clusters with chance 0.5. The following expression is used for calculating the modified Rand index:

$$\begin{aligned} R(C,\hat{C}) &= \frac{\sum_{i>j,\{i,j\}\notin\mathcal{L}}[c_i = c_j \wedge \hat{c}_i = \hat{c}_j]}{2\sum_{i>j,\{i,j\}\notin\mathcal{L}}[\hat{c}_i = \hat{c}_j]} \\ &+ \frac{\sum_{i>j,\{i,j\}\notin\mathcal{L}}[c_i \neq c_j \wedge \hat{c}_i \neq \hat{c}_j]}{2\sum_{i>j,\{i,j\}\notin\mathcal{L}}[\hat{c}_i \neq \hat{c}_j]}. \end{aligned}$$

$\mathcal{L}$ is the set of data-point pairs for which side-information was given to the algorithm. c_i ($\hat{c}_i$) indicates the cluster index of point i according to C ($\hat{C}$).

3.3 Results

We compare our results against Constrained EM (CEM)[3] (Shental et al., 2003). CEM performs expectation maximization (EM) in a Gaussian Mixture Model, where the must-link constraints are enforced via a modified form of the EM update equations, and the cannot-link constraints are enforced via a Markov Random Field net imposed over the hidden cluster assignment variables. It was shown to outperform similar methods (Wagstaff et al., 2001) and (Klein et al., 2002) on a variety of tasks. We also compare SSAP to standard AP in order to validate that SSAP can be used to improve the results of standard AP.

Similarly to AP, SSAP does not take as input the number of clusters to find. Rather, it uses preferences as a tuning parameter. In order to perform the comparison we first run SSAP, and then use the discovered number of clusters as our input to CEM. Fig. 5 demonstrates an example of segmentation result for some of the images in the data set. Fig. 3 describes a quantitative analysis of the 3 algorithms. Each point represents, for a particular amount of instance-level constraints, the average modified Rand index across all 23 images. Each image was subjected to SSAP and CEM with 10 different sets of instance-level constraints. Standard AP does not use the instance-level constraints, and therefore its Rand index is calculated across all point pairs, while that of SSAP and CEM is computed only over data for which no constraints were given, as detailed in section 3.2. Since the number of clusters found by SSAP changes as the amount of constrained information changes, the corresponding AP solutions with the same number of clusters also varies,

[3] code obtained from http://aharon.barhillel.googlepages.com/

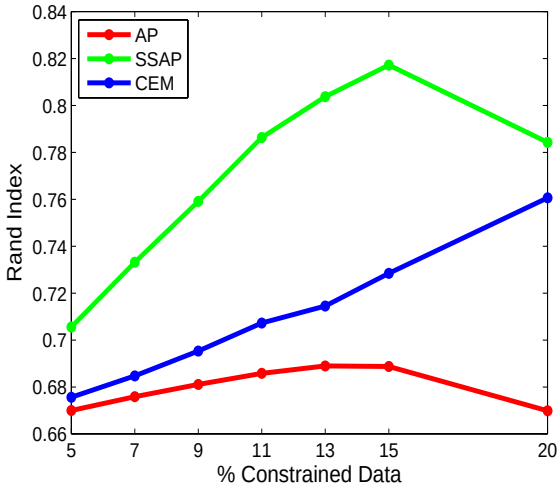

Figure 3: Clustering accuracy as measured by the modified Rand Index for image segmentation.

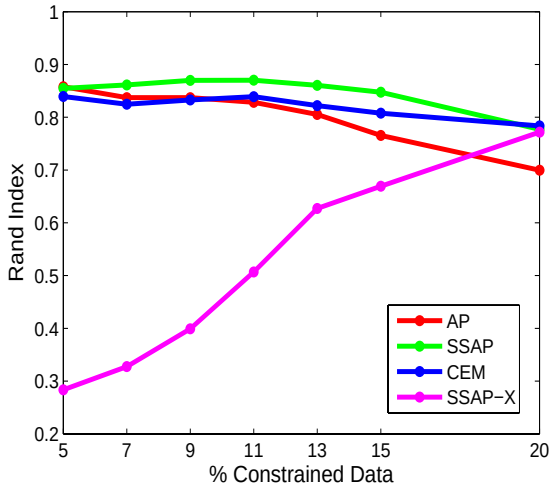

Figure 4: Comparison to SSAP-X for a limited subset of the data, for which comparison can be made, since SSAP-X requires labeled data and assigns labels to all data-points, unlike SSAP and CEM.

and therefore we observe different clustering accuracy for AP for different amounts of constraints although AP does not use these constraints.

We note that comparison between SSAP and CEM is not straight-forward. Although both are clustering algorithms that utilize instance-level constraints, and therefore are most similar in terms of their approach, the similarity measure used by SSAP is given as part of the input and is for the user to decide, while CEM assumes a multi-variate Normal distribution of each cluster and performs maximum-likelihood fitting of the distribution parameters. However, the similarities given to SSAP are based on normalized negative Euclidean distance, and are therefore comparable to a normal distribution assumption.

Since every set of labeled data points can be transformed to a list of instance-level constraints, a natural question is whether SSAP is comparable semi-supervised methods that require labeled data. In particular, we are interested in comparing SSAP to (Xiao et al., 2007) (SSAP-X), which modified the AP algorithm to account for partial labels by contracting all similarly labeled points to a new point, with adjusted similarities, and allowed only the contracted points to serve as exemplars. When making such a comparison, it is important to recall that SSAP may still find clusters that have no labeled information, unlike SSAP-X. The modified Rand index may still be biased for a solution with more clusters. Therefore, we report results only for the subset of experiments where the number of clusters found by SSAP was no more than 1.5 times the clusters found by SSAP-X. This reduces the number of images for which the comparison can be fairly made to (1,2,3,6,10,13,19) images for (5%,7%,9%,11%,13%,15%,20%) of labeled data, respectively. The results are shown in Fig. 4. As can be seen, at the presence of very little labeled data, the instance-level constraints appear to be more helpful then the actual labels, possibly since the number of distinct input labels is less than the number of objects in the image, as well as the fact that the set of potential exemplars is extremely limited and not likely representative of the data.

4 Discussion

We presented a semi-supervised version of affinity propagation for instance-level constraints and demonstrated its applicability to user-interactive image segmentation. This work was motivated by the observation that segmentation results can be improved with very little user interaction. In particular, this interaction can often be restricted to only a subset of the clusters in the image if some clusters are easy to detect in an unsupervised fashion. Therefore, the combination of instance-level constraints with a clustering algorithm capable of finding clusters using side information as well as clusters that are not supervised seems appropriate for this task. The SSAP algorithm we have developed for this task performs well in comparison to similar methods. The ultimate goal would be to build a user-interactive tool that can iteratively refine results. We believe that introducing more structure to the clustering problem, by creating hierarchies for example, can further improve clustering results. In addition, hierarchies can enable a more natural way to define the user-interaction by allowing the user a simple way of indicating how segments should be combined and split apart. We plan to pursue this direction in future work. In addition, using the constraints to

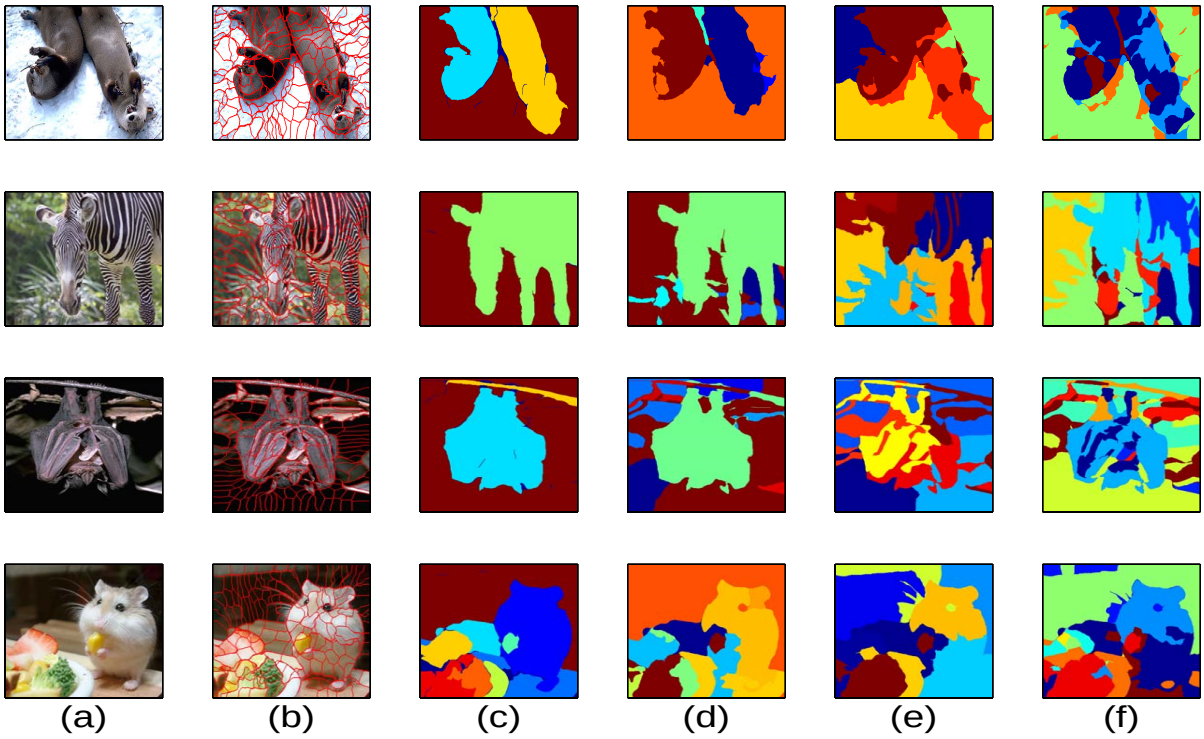

Figure 5: Some examples of image segmentation based on instance-level constraints. Columns from left to right: (a) The original image, (b) superpixels computed for the image, (c) hand-labeling of the super-pixels, (d) SSAP results, (e) AP results, (f) Constrained EM results.

learn a better similarity matrix for AP is also a natural extension of this work.

References

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *Acm sigkdd*. ACM.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *305*(5814), 972 – 976.

Givoni, I., & Frey, B. J. (2009). A binary variable model for affinity propagation. *Neural Computation*.

Klein, D., Kamvar, S. D., & Manning, C. D. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *ICML*.

Kschischang, F., Frey, B. J., & Loeliger, H.-A. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE Transa. Info. Theory*, *47*(2), 498 – 519.

Leone, M., Sumedha, & Weigt, M. (2008). Unsupervised and semi-supervised clustering by message passing: soft constraint affinity propagation. *European Phys.J. B*.

Mori, G. (2005). Guiding model search using segmentation. In *ICCV*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *J. American Stat. Assoc.*, *6*(336), 846–850.

Rother, C., Kolmogorov, V., & Blake, A. (2004). "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, *23*(3), 309–314.

Shental, N., Bar-hillel, A., Hertz, T., & Weinshall, D. (2003). Computing gaussian mixture models with EM using equivalence constraints. In *NIPS*.

Tarlow, D., Zemel, R., & Frey, B. J. (2008). Flexible priors for exemplar-based clustering. In *UAI*.

Wagstaff, K., & Cardie, C. (2000). Clustering with instance-level constraints. In *ICML*.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *ICML*.

Xiao, J., Wang, J., Tan, P., & Quan, L. (2007). Joint affinity propagation for multiple view segmentation. In *ICCV*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In *NIPS*.