

Semi-Supervised Clustering Using Affinity Propagation

Bryan Conroy and Yongxin Taylor Xi

August 24, 2009

1 Introduction

In many fields, the massive size of experimental datasets has required sophisticated machine learning algorithms to better organize and interpret the data. For example, clustering algorithms [6] typically partition a set of N data points into K clusters, given some similarity measure between pairs of data points. Although this has customarily been a completely unsupervised operation, recent studies [5] have explored semi-supervised clustering, in which additional partial labeling information helps to guide the clustering algorithm.

One of the most popular exemplar-based unsupervised clustering algorithms is K -medoids clustering. Each of the K clusters is defined by a set of training points assigned to it, as well as a corresponding data point (exemplar) that represents the cluster center. Given an initial guess of K exemplars, optimization proceeds by alternating between updating the cluster assignment for each data point, and updating the exemplar of each cluster by maximizing the sum of similarities between the exemplar and all points assigned to that cluster.

There are two major drawbacks to the K -medoids approach. First, it is not always possible to intelligently specify the number of clusters K a-priori. Since the derived partition of the data critically depends on the number of clusters, a poorly chosen K could produce nonsensical results. Moreover, clustering is by nature an exploratory tool, and should therefore be able to automatically discover the natural number of clusters in a given dataset. The second drawback is that the algorithm is akin to performing approximate maximum likelihood estimation using the expectation maximization (EM) algorithm. Global convergence of EM is not guaranteed, and the optimized solution will in general depend heavily on the initial choice of cluster exemplars.

Affinity propagation (AP) [2] takes a different approach to clustering. Rather than make hard decisions on the cluster centers at each iteration (as in the M-step of the K -medoids algorithm), soft information about cluster exemplars is propagated through the dataset by way of a message passing algorithm [2]. AP performs the max-sum algorithm on a factor graph [7] model of the data to solve for a good configuration of cluster members. The number of clusters need not be prespecified; instead, the message passing algorithm discovers the number of clusters automatically.

The focus of this paper is on AP, with particular interest in incorporating side information to make it semi-supervised. This study is primarily motivated by two applications: face-image clustering and fMRI voxel clustering. In the following, §2 introduces AP and semi-

supervised clustering, and §3 illustrates the main approaches that we take for semi-supervised AP, followed by the discussion of experiments on face-image clustering, as well as future work.

2 Background

2.1 Affinity Propagation (AP)

Although AP was originally formulated in [2], we instead follow the equivalent derivation given in [4], since it has been shown to be easier to extend to other formulations. We assume that we are given a training set of data points $D = \{x_i \in \mathbb{R}^n\}_{i=1}^N$ and a pairwise similarity measure $s(i, j)$ between pairs of data points x_i and x_j , $i, j = 1, \dots, N$. The similarity $s(i, j)$ could, for example, represent the log likelihood of x_i conditioned on its exemplar x_j . In the case of mixture of Gaussians, $s(i, j) = -\|x_i - x_j\|^2$, corresponding to the log likelihood of x_i given its mean x_j . The self-similarity $s(j, j)$ acts as an input preference that x_j is an exemplar: a larger $s(j, j)$ relative to the self-similarities of other data points increases the likelihood that x_j is selected as an exemplar. Without prior knowledge, the self-similarities can be set to a fixed constant.

Let $C = [c_{ij}]$ denote a $N \times N$ matrix of hidden binary-valued elements, whose i^{th} row encodes the cluster assignment for x_i . In particular, $c_{ik} = 1$ for some $k \in \{1, \dots, N\}$ indicates that x_k is the exemplar for x_i . AP seeks to maximize the sum of similarities between data points and their exemplars, $E(c) = \sum_{i,j=1}^N S_{ij}(c_{ij})$, where $S_{ij}(c_{ij}) = s(i, j)$ if $c_{ij} = 1$, and 0 otherwise. To account for invalid configurations on C , the objective function is modified by two sets of additive constraint functions $\{I_i\}_{i=1}^N$ and $\{E_j\}_{j=1}^N$, given by:

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

$$E_j(c_{1j}, \dots, c_{Nj}) = \begin{cases} -\infty & , \text{ if } c_{jj} = 0 \text{ and } \exists i \neq j : c_{ij} = 1 \\ 0 & , \text{ otherwise} \end{cases}$$

The function I_i operates on the i^{th} row of C , enforcing that x_i is assigned to only one cluster. The function E_j operates on the j^{th} column of C , ensuring that x_j chooses itself as an exemplar if another data point chooses x_j as its exemplar.

The optimal configuration C then maximizes the sum:

$$S(C) = E(C) + \sum_{i=1}^N I_i(c_{i1}, \dots, c_{iN}) + \sum_{j=1}^N E_j(c_{1j}, \dots, c_{Nj}), \quad (1)$$

AP optimizes for C by the max-sum algorithm on a particular kind of graphical model of the data, called a factor graph [7].

Factor graphs were introduced in [7] as a useful way to express a factorization of a global function g of many variables into smaller local functions. For example, suppose g represents the joint distribution of a set of random variables x_1, \dots, x_N , and that it factors into a set of m local functions $f_j, j = 1, \dots, m$:

$$g(x_1, \dots, x_N) = \prod_{j=1}^m f_j(X_j),$$

where X_j is a subset of $\{x_1, \dots, x_N\}$. The graph for g contains a variable node for each variable $x_i, i = 1 \dots, N$, and a factor node for each local function $f_j, j = 1 \dots, m$. An edge connects the factor node representing local function f_j with each of the variables in the set X_j . Typically, variable nodes are represented by circular nodes, and factor nodes are represented by shaded square nodes.

Factor graphs can be used to derive message passing algorithms that compute the sum-product and max-product algorithms on g [7]. By taking the logarithm of g , the max-product algorithm can be extended to the max-sum algorithm. This approach is taken in AP to optimize C based on maximization of $S(C)$ in (1). A factor graph for AP is shown in Figure 1(a).

Messages are only passed between adjacent nodes in the graph, and the five types of messages, S_{ij} , α_{ij} , β_{ij} , η_{ij} , and ρ_{ij} , are summarized in Figure 1(b). Although these messages depend on the value of c_{ij} , the difference between message values can be sent since c_{ij} is a binary variable, i.e., $\rho_{ij} = \rho_{ij}(1) - \rho_{ij}(0)$. In addition, only three of these messages need to be passed, as β_{ij} and η_{ij} can be expressed in terms of the remaining three messages. Based on the update rules for the max-sum algorithm, the remaining messages are given by [4]:

$$\begin{aligned} S_{ij} &= s(i, j) \\ \rho_{ij} &= s(i, j) - \max_{k \neq j} (s(i, k) + \alpha_{ik}) \\ \alpha_{ij} &= \begin{cases} \sum_{k \neq j} \max[\rho_{kj}, 0] & , \text{ if } i = j \\ \min \left[0, \rho_{jj} + \sum_{k \notin \{i, j\}} \max[\rho_{kj}, 0] \right] & , \text{ if } i \neq j \end{cases} \end{aligned}$$

In AP, ρ_{ij} and α_{ij} are called the responsibility and availability messages, respectively. The responsibility message transfers information from x_i to x_j about how well-suited x_j is as an exemplar for x_i . In return, x_j transfers information to x_i through the availability message, which informs x_i on the merits of x_j as an exemplar, given information x_j has received from other data points.

When AP converges, each of the configuration variables c_{ij} is determined by adding all messages incoming to node c_{ij} and setting $c_{ij} = 1$ if the sum is positive, and 0 otherwise.

2.2 Semi-supervised Clustering

As shown in the previous sections, AP is an unsupervised clustering algorithm. In semi-supervised clustering, a small amount of label information is available concerning pairwise (must-link or cannot-link) constraints between data points. This information provides supervision and thus can be used to guide the clustering process. There are two main approaches in semi-supervised clustering: similarity-adapting methods, and search-based methods [5]. In similarity-adapting methods, a preprocessing step adapts the similarity measure between data points to facilitate the compliance of pairwise constraints. After that, an existing unsupervised algorithm can be used to cluster with the new similarity measure. In search-based methods, we modify the clustering algorithm such that the label information is used to bias the search for an appropriate clustering. One can introduce a cost function to penalize constraint violations, or require hard constraints to be satisfied during the cluster assignment process.

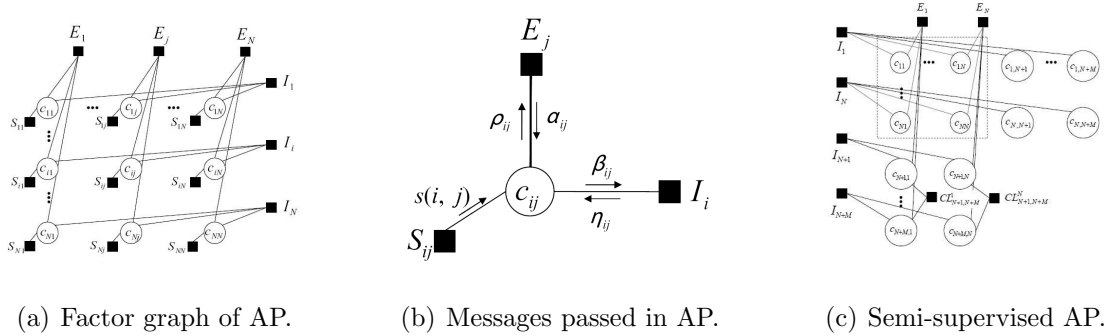


Figure 1: Factor graphs and messages for AP; A semi-supervised variate of AP.

2.3 Semi-supervised Affinity Propagation

We ask the question how one can turn affinity propagation to a successful semi-supervised learning algorithm. Very recently, Givoni [3] describe a search-based method which augments the data nodes in AP with fictitious ‘meta-points,’ or MTPs. Each MTP is added to a transitive closure of the must-link constraints, and can therefore explicitly enforce the must-link and cannot-link constraints. As shown in Figure 1(c), this method introduces $2NM$ additional binary nodes to the original factor graph of AP, where N is the total number of points, and M is the number of transitive closures (we denote this number as $|M| + |C|$ in later sections). N additional function nodes CL^1 to CL^N are introduced for the purpose of cannot-link constraints. Although this approach has been shown to be successful in the application of image segmentation, the additional nodes and functions make the computation more cumbersome.

We hope to design a more efficient algorithm for semi-supervised AP. As shown in Section 2.2, there are two possible approaches for this: one, by adapting the similarity measure; two, by changing the penalty functions $\{E_j\}_{j=1}^N$ to penalize any violation of the constraints.

This paper mainly takes the first approach. However, before elaborating, we first briefly discuss why the second approach may only provide a trivial clustering solution. For example, by modifying the penalty function of E_j defined in (2), the new objective explicitly makes all the constraints satisfied:

$$E_j(c_{1j}, \dots, c_{jN}) = \begin{cases} -\infty & , \text{ if } c_{jj} = 0 \text{ and } \exists i \neq j : c_{ij} = 1 \\ -\infty & , \text{ if } c_{uj} + c_{vj} = 1 \text{ and } (u, v) \text{ a must-link pair.} \\ -\infty & , \text{ if } c_{pj} + c_{qj} = 2 \text{ and } (p, q) \text{ a cannot-link pair.} \\ 0 & , \text{ otherwise} \end{cases} \quad (2)$$

As pointed out in [3], solving for such problems directly without propagating to the neighboring points may lead to a trivial solution, where all the must-link points computed from transitive closure will be clustered together, possibly ignoring their neighbors. Furthermore, it introduces an integer programming problem which might be difficult to solve efficiently.

Therefore in this paper we explore the other approach, i.e. similarity propagation methods for an efficient semi-supervised AP algorithm. Rather than modifying the similarity measure at the very beginning, we design a negative feedback framework to supervise similarity propagation throughout the iterations of affinity propagation.

3 Proposed Algorithms for Semi-Supervised AP

The fundamental problem is: given N data points $\{x_i\}_{i=1}^N$, cluster them according to their similarity and partial pairwise constraints. Let M be the set of must-link pairs, and C be the set of cannot-link pairs.

3.1 Method 1: Similarity Update Using a Gaussian Kernel

The ultimate goal of the semi-supervised AP is to minimize the following objective (\vec{c}_i is the i th row of matrix c):

$$\begin{aligned} \min \quad & L(c) = - \sum_{c_{ij}=1} S(i, j) \\ \text{S.t.} \quad & \forall (i, j) \in M, \vec{c}_i = \vec{c}_j \\ & \forall (i, j) \in C, \vec{c}_i \neq \vec{c}_j \\ & \text{Propagation of constrained points to their neighbors.} \end{aligned}$$

If we only have an $N \times N$ similarity matrix s at hand, then we need some mechanism to guide the update of similarity measure for a better fit of the pairwise constraints.

A heuristic method is to locally change the similarity between the K nearest neighbors of x_i and K nearest neighbors of x_j if constraint (i, j) is not satisfied. Let

$$\begin{aligned} s(p, q) &= s(p, q)_{\text{old}} + \delta \mathbf{1}\{p \in N_i \text{ and } q \in N_j\}, \text{ if } (i, j) \in M \text{ and } \vec{c}_i \neq \vec{c}_j \\ &= s(p, q)_{\text{old}} - \delta \mathbf{1}\{p \in N_i \text{ and } q \in N_j\}, \text{ if } (i, j) \in C \text{ and } \vec{c}_i = \vec{c}_j \end{aligned}$$

where δ is a small positive value and N_i is the set of K neighbors of point x_i including itself. This method increases/decreases the distances of not-yet-satisfied must-link/cannot-link pairs (together with K neighbors).

The approach above is easy to implement and it works in some applications. However, the naive step size δ is chosen to be the same for the K nearest neighbors. This might cause undesirable discontinuous wrapping to the original distance space.

To address this problem, we could use the following similarity update on points (y, z) , parametrized by α :

$$s(y, z) = s(y, z)_{\text{old}} + \sum_{(i,j) \in M, \vec{c}_i \neq \vec{c}_j} \alpha_{ij} \mathcal{G}(y, z, x_i, x_j) - \sum_{(i,j) \in C, \vec{c}_i = \vec{c}_j} \alpha_{ij} \mathcal{G}(y, z, x_i, x_j) \quad (3)$$

where $\alpha_{ij} \geq 0$, $\|\alpha\|_p \leq \text{const}$. Here

$$\mathcal{G}(y, z, x_i, x_j) = (G(s(y, x_i), s(z, x_j)) + G(s(z, x_i), s(y, x_j)))/2 \quad (4)$$

where G is the Gaussian kernel such that $G(a, b) = e^{-(a^2+b^2)/\sigma^2}$.

The parameterized similarity update has in all $|M| + |C|$ parameters, i.e. $\{\alpha_{ij} : (i, j) \in M \cup C\}$. A naive way to set α is to make them all zero at the beginning. Then after each iteration of affinity propagation, check all the pairwise constraints, then set α_{ij} to be a constant small value if (i, j) constraint is still violated.

Equation (3) essentially draws closer a must-link pair and their neighbors, and at the same time pushes apart a cannot-link pair and their neighbors. Therefore, the proposed method will interactively learn a set of parameters from the not-yet-satisfied constraints in order to yield a better similarity measure for affinity propagation. Affinity propagation will take care of the label assignments, while the current procedure to update the similarities can facilitate and expedite the process.

In practice, motivated by [9], we make the variance of the Gaussian kernel adaptive to the size of the local clusters: $G(s(y, x_i), s(z, x_j)) = \exp(-s(y, x_i)^2/\sigma_i^2 - s(z, x_j)^2/\sigma_j^2)$, where σ_i is the distance (negative similarity) from point x_i to its exemplar.

There are a few drawbacks to this approach. First, the Gaussian is isotropic, and therefore information from labeled points is distributed to its neighbors in a symmetric fashion. If the clusters are not isotropic, this approach may be inadequate. Second, if the similarity measure gets updated too many times, it may overfit to the available constraints and may not generalize well to other data points. However, if we are not given any other information (e.g. data in original space) besides the similarity of each pair, this method could be used to incorporate partial supervision.

3.2 Method 2: Similarity Metric Learning

Now consider a different scenario: we are given points in their original space, being asked to compute a good similarity measure for them with partial supervision. Then we have the freedom to parameterize a global similarity measure for all points, e.g. let $s_D(x_i, x_j) = x_i^T D x_j$ where $x_i, x_j \in R^p$ are unit norm vectors and D is a diagonal positive semi-definite matrix. Note that, with D equal to the identity matrix, $s_D(x_i, x_j)$ measures the correlation between datapoints x_i and x_j . Optimizing over parameter D and labels c_{ij} for some loss function may simultaneously yield a good similarity measure and cluster assignments. This approach is motivated by [1]. We propose maximizing the following objective function:

$$\max \sum_{c_{ij}=1} s_D(i, j) - \lambda \left(\sum_{(i,j) \in M} (s_{max} - s_{ij}) \mathbf{1}\{\vec{c}_i \neq \vec{c}_j\} - \sum_{(i,j) \in C} (s_{ij}) \mathbf{1}\{\vec{c}_i = \vec{c}_j\} \right) \quad (5)$$

such that $\|\text{diag}(D)\|_2 = 1$ and s_{max} is set to be a large positive constant.

Problem (5) optimizes for a weighted version of clustering performance and constraint compliance. The first term encourages a good clustering assignment for all the data points, and the second term penalizes the violation of constraints in M and C . Also, constraints on distant must-link pairs and nearby cannot-link pairs are weighed more heavily. (see [1] for a justification.)

While affinity propagation works toward a better clustering performance (the first term), the following method can be used to update the similarity measure for better constraint compliance:

$$\max I(D) = \sum_{(i,j) \in M} s_D(i, j) \mathbf{1}\{\vec{c}_i \neq \vec{c}_j\} - \sum_{(i,j) \in C} s_D(i, j) \mathbf{1}\{\vec{c}_i = \vec{c}_j\} \quad (6)$$

such that $\|\text{diag}(D)\|_2 = 1$.

Let $D = \text{diag}(d_1, d_2, \dots, d_n)$ where $d_k > 0, \forall k$ and $\sum_k d_k^2 = 1$. The derivative of (6) w.r.t d_k can be computed by

$$\frac{\partial I}{\partial d_k} = \sum_{(i,j) \in M, \vec{c}_i \neq \vec{c}_j} (x_i)_k (x_j)_k - \sum_{(i,j) \in C, \vec{c}_i = \vec{c}_j} (x_i)_k (x_j)_k \quad (7)$$

where $()_k$ denotes the k th element of a vector.

To increase I , we can update d (diagonal vector of D) in the gradient direction $\frac{\partial I}{\partial d}$ while keeping all d_k positive, then re-normalize d back to the unit l_2 norm.

Notice that this method only takes into consideration the label information that the clustering has gotten wrong. We discuss another variant that utilizes all of the label information at each iteration in the Future Work section.

3.3 Method 3: Boost the Similarity Metric

The previous section uses the gradient ascent of the objective function to update the similarity measure. In this section, we use boosting as a supervised learning tool to simultaneously enhance the similarity metric and clustering. As an augment to the previous approach, it is able to learn a positive semi-definite metric instead of a diagonal one. Also, it offers a new perspective on formulating the clustering problem.

Clustering is the assignment of a set of n points into K subsets (i.e. clusters) so that points in the same cluster are similar in some sense. There are two types of clustering: hard clustering and fuzzy clustering. In hard clustering, each point is assigned to a single cluster. Fuzzy clustering, on the other hand, assign a probability distribution of different clusters to each point. For both cases, we can use a group cluster matrix $X \in R^{n \times K}$ s.t. $\forall i \sum_j X_{ij} = 1$ to fully describe cluster membership, with an extra condition $X_{ij} \in \{0, 1\}$ for hard clustering. Note that fuzzy clustering can be easily transformed to a hard clustering by taking maximum over rows, but not vice versa.

An important observation is that matrix X always has rank K for a K cluster problem. Therefore $G = XX^T$ also has rank K , which is usually a lot fewer than the size of matrix G . $G(i, j) = \sum_{k=1}^K X_{ik} X_{jk}$ is then the probability of point i and j being in the same cluster, given the independence among clusters. In semisupervised clustering, we are given two sets of constraints: must links M and cannot links C . Therefore $\{(i, j) \in M \cup C\}$ serve as labeled points: $G(i, j) = 1$ if $(i, j) \in M$ or 0 if $(i, j) \in C$.

To formulate a boosting problem, the input of the classifier is a pair of points (x_i, x_j) and the output label $Y(i, j) = 1$ if x_i and x_j are classified as in the same cluster and $Y(i, j) = -1$ otherwise. We are given a set of labels in advance: $y_{ij} = 1$ for $(i, j) \in M$ and $y_{ij} = -1$ for $(i, j) \in C$. We define a set of weights on these labeled points: $\{w_{ij} : (i, j) \in M \cup C\}$. At the t th iteration, a new distance metric is learned w.r.t these weights. Affinity propagation takes in the new pairwise similarity to produce a basic clustering, which can be easily transformed to binary square matrix Y_t . Then we update weights based on Y_t , i.e. increase weight of points that are classified wrongly, and decrease the weight of points that are correct. The final aggregated classification matrix $Y = \sum_{t=1}^T \alpha_t Y_t$, where larger α_t indicates better classification accuracy of Y_t . Finally we transform Y to a clustering probability matrix G then decompose $G \approx XX^T$ to obtain group cluster matrix X .

Now we iterate each step in greater detail. At the first iteration, the weights are set to be uniform: $w_{ij} = \frac{1}{|M|+|C|}$. Let the similarity metric be $s_D(x_i, x_j) = x_i^T P x_j$ where $x_i, x_j \in R^p$ are unit norm vectors and P is a positive semi-definite matrix. The initial P is set to be the identity matrix. At each boosting iteration, we solve the following problem

$$\begin{aligned} \max_{\Delta} \quad & \sum_{(i,j) \in M} w_{ij} x_i^T \Delta x_j - \sum_{(i,j) \in C} w_{ij} x_i^T \Delta x_j \\ = \Delta \odot \quad & \left(\sum_{(i,j) \in M} w_{ij} x_i x_j^T - \sum_{(i,j) \in C} w_{ij} x_i x_j^T \right) \\ = \text{trace} \quad & (U S U^T \Delta) \end{aligned}$$

where $U S U^T$ is the SVD of symmetric matrix $(\sum_{(i,j) \in M} w_{ij} x_i x_j^T - \sum_{(i,j) \in C} w_{ij} x_i x_j^T)$. Therefore $\Delta = U \tilde{S} U^T$ where $\tilde{S}(i, i) = 1$ if $S(i, i) > 0$ and 0 otherwise. The new positive semi-definite matrix is $P = P + \epsilon \Delta$, which is used by affinity propagation in the next step.

Affinity propagation takes in the adjusted similarity, and returns clusters of points with an exemplar for each cluster, which uniquely determines Y_t . That is, if x_i and x_j are clustered together, $Y_t(i, j) = 1$. Otherwise $Y_t(i, j) = -1$. If the classification error e_t of labeled points is above 50%, discard Y_t , further adjust the similarity by $P = P + \epsilon \Delta$ then run affinity propagation again. Otherwise, the coefficient for the t th clustering matrix is $\alpha_t = \frac{1}{2} \ln \frac{1-e_t}{e_t}$. The final step of each iteration is updating the weight on each label,

$$w_{ij} = e^{-\alpha_t y_{ij} Y_t(i,j)}$$

The aggregated clustering matrix $Y = \sum_{t=1}^T \alpha_t Y_t$ is the final classification for each pair of points to be in the same cluster or not. The sign of $Y(i, j)$ shows such classification, while the magnitude of it means the confidence of such classification. Boosting theory says that the classification error of labeled points decreases exponentially if e_t is strictly less than half at each iteration. Empirically the test error will decrease along with the training error even after the training error reaches 0.

To obtain G then decompose it to obtain group cluster matrix X such that $G \approx X X^T$, two approaches can be used. The first one set G to be the thresholded Y at 0 then computes the SVD of G . Since $Y_{ij} < 0$ means that the majority vote for i and j to be in the same cluster is negative, we threshold them to 0. G is now a non-negative matrix. Computing the SVD of G leads to orthogonal X , yet X is not necessarily non-negative. Finally we take the absolute value of X and assign the index of the maximum of the i th row X_i to the label of the i th point.

The second approach first maps Y into G such that $0 \leq G_{ij} \leq 1$, then conduct the nonnegative matrix factorization: $G \approx X X^T$ s.t. $X \geq 0$ element-wise. An advantage of this approach is the preservation of the confidence of classification. In addition, every entry of X will be non-negative. However, to use this approach, one has to know or estimate the number of clusters. Furthermore, it might suffer from local minima due to the non-convexity of the optimization problem.

4 Experiment

We applied our two semi-supervised algorithms outlined in §3 to a face-image clustering problem – given a database of face images, cluster the images based on identity. This problem fits well into the semi-supervised clustering problem, since some of the faces may be tagged by identity. These tags imply a set of link constraints: a must-link constraint for each pair of data points that are tagged the same, and a cannot-link constraint for each pair of data points that are tagged as different identities.

We used two freely-available face databases: the Yale face database ¹, and AT&T’s ORL face database ². The Yale face database contains 11 example images for each of 15 individuals, while the ORL database is larger, with 40 individuals and 10 examples per individual. In both databases, the images were acquired under different lighting configurations, with and without glasses, and different facial gestures. The example images for 15 individuals from the Yale face database are shown in Figure 2(a), and 11 images for one individual is shown in Figure 2(b).

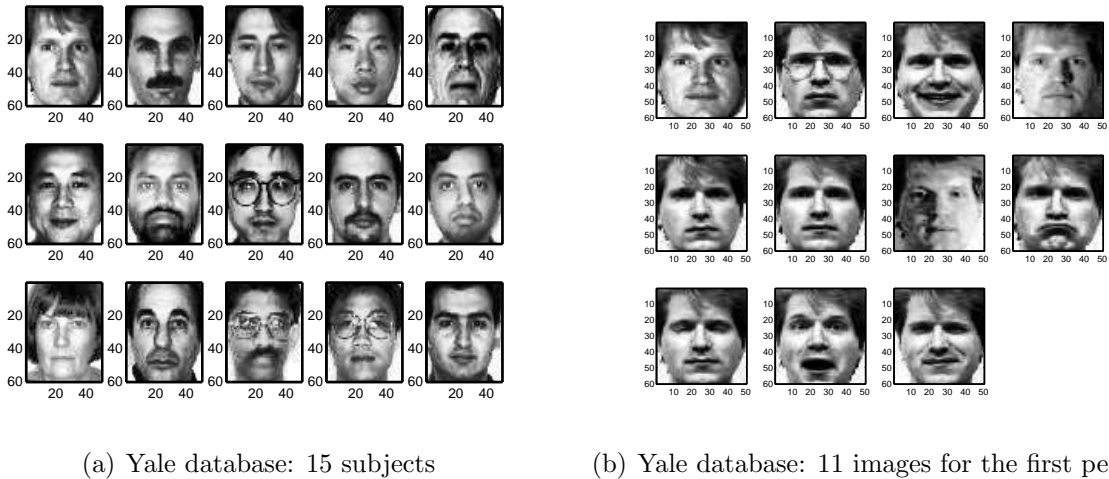


Figure 2: Yale database illustration.

As a preprocessing step, Principal Component Analysis (PCA) was used to reduce the size of the data and hopefully remove irrelevant dimensions. Given a matrix $X \in \mathbb{R}^{d \times N}$ whose columns correspond to face images, PCA computes a $m \leq d$ dimensional approximate representation $X \approx U_m \Lambda_m V_m^T$, where $U \in \mathbb{R}^{d \times m}$ and $V \in \mathbb{R}^{N \times m}$ are matrices with orthogonal columns, and $\Lambda_m \in \mathbb{R}^{m \times m}$ is a diagonal matrix of singular values. This approximation implies an m -dimensional coordinate representation for the set of face images: $X_{PCA} = U_m^T X = \Lambda_m V_m^T$. The matrix X_{PCA} was input as the data matrix to our algorithms.

We had a parameter – the percentage of constrained data, which is the percentage of data labeled a-priori. To test each value of percent labeled data, we ran 25 uniformly random initializations to collect labels. Then we averaged the results over those 25 runs.

¹Yale: <http://cvc.yale.edu/projects/yalefaces/yalefaces.html>

²ORL: <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

Selecting an objective method to evaluate the clustering performance is important. When the ground truth label information is not available, reconstruction error is often used as a measure of exemplar-based clustering accuracy, which is defined as the sum of errors incurred by approximating each data point by its cluster exemplar. This does not give much insight into how close the clustering is to the true data partition, since the clustering with minimal reconstruction error is the trivial one – assign each data point to its own cluster. Since we had the true face-image cluster labels, we instead chose to use the adjusted Rand index [Rand71] to evaluate the clusterings produced by affinity propagation and semi-supervised affinity propagation. Given a set of data $X = \{x_1, \dots, x_N\}$, the adjusted Rand index R between the true label information $L = \{l_1, \dots, l_N\}$ and a second set of labels $\bar{L} = \{\bar{l}_1, \dots, \bar{l}_N\}$, generated from one of the clustering algorithms, is defined as:

$$R = \frac{2(ab - cd)}{((a + d)(b + d) + (a + c)(b + c))}$$

where

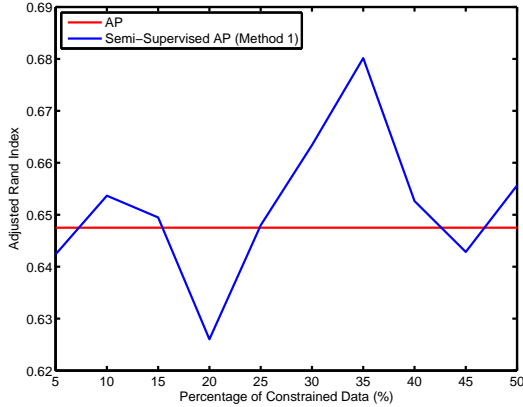
$$\begin{aligned} a &= \Pr [\text{two elements in } X \text{ are in the same set in both } L \text{ and } \bar{L}] \\ b &= \Pr [\text{two elements in } X \text{ are in different sets in both } L \text{ and } \bar{L}] \\ c &= \Pr [\text{two elements in } X \text{ are in the same set in } L \text{ and in different sets in } \bar{L}] \\ d &= \Pr [\text{two elements in } X \text{ are in different sets in } L \text{ and in the same set in } \bar{L}] \end{aligned}$$

The adjusted Rand index R is equal to the corrected-for-chance probability that the two clusterings agree on whether or not two randomly selected elements from X belong in the same cluster. Thus, $R \in [0, 1]$, with $R = 1$ implying that the two clusterings partition the data the same.

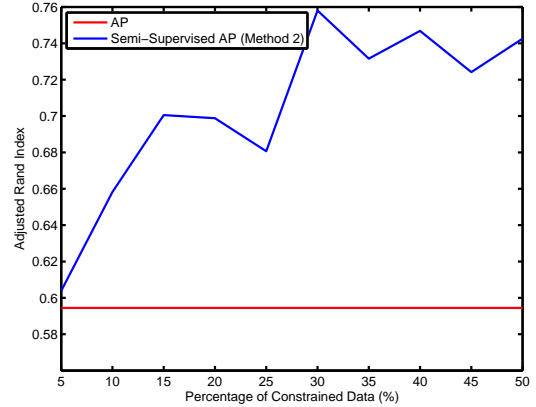
Since the semi-supervised algorithm takes advantage of side information that affinity propagation does not have access to, we made the comparison fair in the following way: when computing the adjusted Rand index on the semi-supervised clustering result, we excluded data points that were labeled a-priori.

Figure 3 plots the adjusted Rand index for both affinity propagation and semi-supervised affinity propagation using the two methods. Although the first method based on altering the similarities using a Gaussian kernel does not work well for face clustering, as shown in Figure 3(a), the second method based on metric learning effectively utilizes the label information to improve the clustering performance. When applying the MTP algorithm proposed in [?], we also noticed idiosyncratic results similar to the first method.

Why is the first approach (and the MTP algorithm) unsuccessful in face clustering? The mechanism of these approaches is to draw closer the unsatisfied must-link pairs, and expel the cannot-link pairs. This is accomplished by encouraging the neighbors (in the original feature space) of these points to either share cluster exemplars or differ in cluster exemplars. If we are given good discriminative features in the first place, i.e. data points that already appear to be well separated, then the first approach as well as Givoni’s method [3] should work well. However, for high dimensional datasets such as face images, neither the original pixel space nor the reduced PCA space partitions the faces well. As a result, propagating label information from a constrained datapoint to its neighbors may actually pass information across different clusters.



(a) Rand Index for the Gaussian method



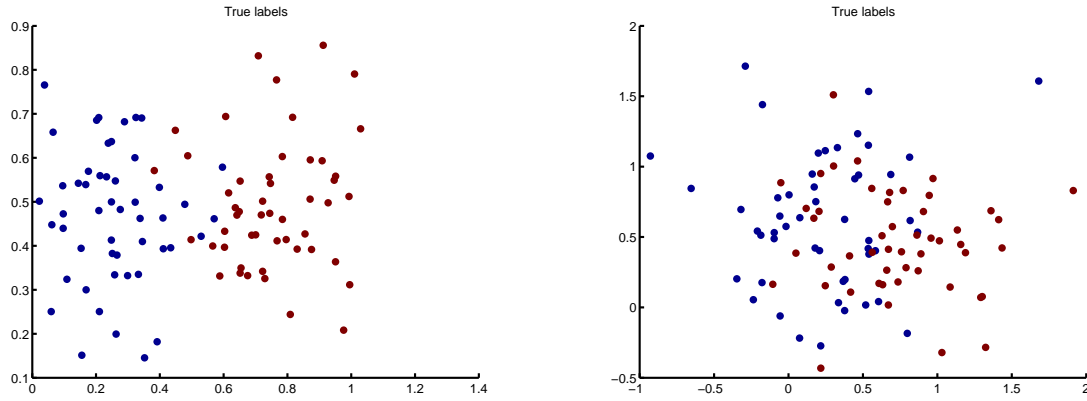
(b) Rand Index for the Metric Learning method

Figure 3: Clustering performances of semi-supervised Affinity Propagation.

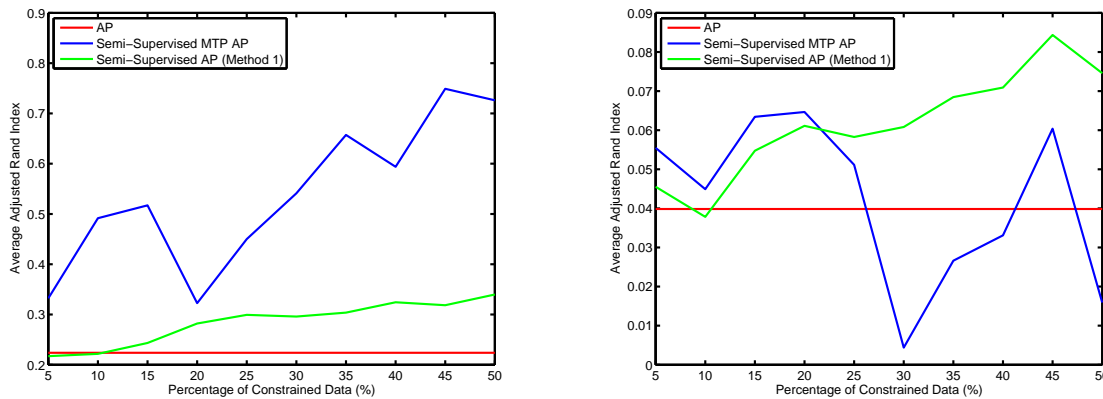
To validate this claim, we tested both the Gaussian similarity method and the MTP algorithm on a simple synthetic data example. We generated data from a mixture of two Gaussians, and varied the separation between the two classes. Figure [...] shows the case where there is moderate separation between the two classes, while Figure [...] shows the case where there is significant overlap between the two classes. While both algorithms effectively utilize the partial label information when the clusters are separated, in this example, the side information adversely influences the MTP algorithm when there is overlap between the clusters. In other simulations, we see similar effects on the Gaussian similarity method.

In contrast, our proposed metric learning method performs well on face clustering because a new distance metric is learned for the original data from the partial labels. In this sense, it performs feature selection and clustering simultaneously. The converged diagonal values of the matrix D inform the relative importance of the different features of the data. For example, Figure 5(a) records 50 converged diagonal entries of D (averaged from 25 random runs), corresponding to the first 50 principal components of the face dataset. Figure 5(b) shows the components with 5 largest weights, and Figure 5(c) shows the least 5.

An important remark about Figure 5 is that the largest principal components do not necessarily achieve the highest weights. This is because leading principal components seek to maximize the variance of the data, so they may not necessarily serve as good discriminative features between clusters. Actually, as shown in Figure 5(c), the second and third principal components both received some of the lowest weights. The second principal component (left, first row) seems to capture left/right illumination variation and the third principal component (middle, second row) appears to capture the large hair variance. The most weighted components in Figure 5(b) are instead the 7, 12, 13, 9, and 6th components, which appear to be more discriminative. The algorithm therefore allows us to identify the set of informative features and has the potential of effective dimensionality reduction.

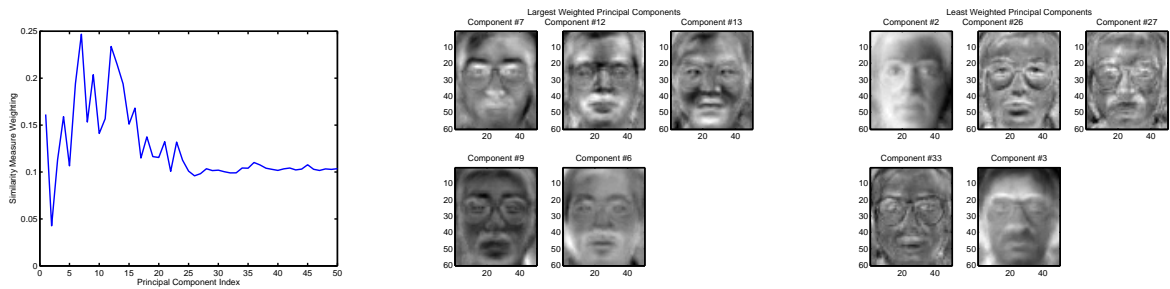


(a) 2-class clustering with low overlap between classes (b) 2-class clustering with large overlap between classes



(c) AP and Semi-Supervised AP results on data in Figure 4(a) (d) AP and Semi-Supervised AP results on data in Figure 4(b)

Figure 4: Synthetic Clustering Example. The Semi-Supervised MTP AP algorithm is that proposed in [3]



(a) Metric learning: converged diagonal values of D (b) Largest weighted components (c) Least weighted components

Figure 5: Analysis of the metric learning approach.

5 Future Work

5.1 Learning a Positive Semi-Definite Similarity Matrix

The success of the second method is due to its ability to learn a weighted set of good features for the dataset from the partial label constraints. However, under the current model, the

matrix D was constrained to be a diagonal matrix for computational convenience. This allows for features to be scaled, but disallows interactions between the features. However, for cases where the features are dependent in some way, we should generalize this matrix D to be any symmetric positive semi-definite (PSD) matrix:

$$D = O\Sigma O^T \quad (8)$$

Here, Σ is a diagonal non-negative matrix, and the matrix $O \in \mathbb{R}^{N \times N}$ is orthonormal. One approach to solve for O is by optimization on the Stiefel manifold of orthonormal matrices. However, this approach may be slow when the number of data points, N , is large. Alternatively, we can employ a greedy algorithm by decomposing O as a product of Givens rotations:

$$O = \prod_{i=1}^K G_i \quad (9)$$

Here, each Givens rotation G_i is uniquely specified by three parameters: two axes and a rotation angle. At iteration k , the best G_k is selected greedily, given the previously optimized rotation matrices $\{G_1, \dots, G_{k-1}\}$. Although any orthonormal matrix O can be decomposed in the form (9) with $K \leq \frac{N(N-1)}{2}$; in practice, K can be set to a smaller number to limit the number of degrees of freedom. This approach was successfully used recently for the purpose of covariance estimation in [?].

We plan to apply this approach to face-image clustering on the ORL face database and believe that it will improve clustering performance over the case when D is constrained to be a diagonal matrix. Whereas some of the principal components of the Yale face database were discriminative between identities (see Figure [...]), none of the principal components of the ORL face database appear to be individually discriminative (see Figure [...] for the top 50 principal components). This is a common fault of principal component analysis in classification: PCA finds components that represent the data well, but may not find components of interest to the classification problem. By generalizing D to allow for linear combinations of these components, the partial label information may allow us to find more discriminative features that partition the data well for clustering.

5.2 Connection to Boosting

In supervised learning, boosting updates the weights on the labeled examples according to the current classification performance on these points. Difficult examples that are often misclassified are weighted more, which effectively increases the attention they receive in later boosting rounds. In analogy to our second method, (5) is an extreme example of the boosting approach. At each iteration, link constraints that are "misclassified" by the clustering algorithm receive a weight of 1, while those that are correctly classified receive a weight of 0.

One pitfall of our current approach is that only a portion of the labeled information is used at each iteration (e.g., the constraints that the clustering algorithm violates). Alternatively, the satisfied constraints can also obtain nonzero (yet small) weights, just like in boosting. This is equivalent to replacing each indicator function in (5) with an exponential function that decreases monotonically with the number of times the constraint is satisfied by the

clustering algorithm. This allows us to dynamically utilize all of the label information at each iteration. The connection to boosting therefore merits further investigation.

5.3 Soft-Constraint Affinity Propagation

Recently, [8] proposed a modified version of affinity propagation that is based on statistical-physics based clustering techniques. In the paper, a simplified version of the algorithm is derived and possibilities of time- and memory-efficient implementations are discussed. In the original AP, a hard constraint is: whenever a point is chosen as an exemplar by some other point, it is forced to be also its own self-exemplar. In the soft constraint version, this hard constraint is replaced by a soft regularization term in the objective function which penalizes the total number of exemplars. The author argues about several disadvantages of the hard constraints and shows an improvement using the new approach. We would like to examine this approach as well in the future work.

6 Conclusion

References

- [1] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, New York, NY, USA, 2004. ACM.
- [2] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
- [3] I. Givoni and B. Frey. Semi-supervised affinity propagation with instance-level constraints. In *12th Artificial Intelligence and Statistics*, 2009.
- [4] Inmar E. Givoni and Brendan J. Frey. A binary variable model for affinity propagation. *Accepted to Neural Computation*.
- [5] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semisupervised clustering: a brief survey. In *in A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.
- [6] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [7] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.
- [8] Michele Leone and Martin Weigt. Unsupervised and semi-supervised clustering by message passing: Soft-constraint affinity propagation, 2007.

- [9] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2004.