LEIBNIZ CENTER FOR RESEARCH IN COMPUTER SCIENCE
TECHNICAL REPORT 2003-43

Computing Gaussian Mixture Models with EM using Equivalence Constraints

Noam Shental, Aharon Bar-Hillel, Tomer Hertz and Daphna Weinshall
School of Computer Science and Engineering and the Center for Neural Computation
The Hebrew University of Jerusalem, Jerusalem, Israel 91904
Email: {fenoam,aharonbh,tomboy,daphna}@cs.huji.ac.il

Abstract

Gaussian mixture models for density estimation are usually estimated in an unsupervised manner, using an Expectation Maximization (EM) procedure. In this paper we show how *equivalence constraints* can be incorporated into this procedure, leading to improved model estimation and improved clustering results. *Equivalence constraints* provide additional information on pairs of data points, indicating if the points arise from the same source (positive constraint) or from different sources (negative constraint). Such constraints can be gathered automatically in some learning problems, and are a natural form of supervision in others. We present a closed form EM procedure for handling positive constraints, and a Generalized EM procedure using a Markov network for the incorporation of negative constraints. Using publicly available data sets, we demonstrate that incorporating *equivalence constraints* leads to considerable improvement in clustering performance, and that our algorithm outperforms all available competitors.

Keywords: semi-supervised learning, equivalence constraints, clustering, EM, Guassian mixture models

1 Introduction

Gaussian Mixture Models (GMM) for density estimation are popular for two main reasons: they can be reliably computed by the efficient Expectation Maximization (EM) algorithm [1], and they provide a generative model for the way the data may have been created. The second property in particular makes for their common use for unsupervised clustering, where typically the Gaussian components of the GMM model are taken to represent different sources. This use is common because most other clustering algorithms are not generative, and therefore cannot provide predictions regarding previously unseen points.

When used for clustering in this way, the underlying assumption - i.e., that the density is comprised of a mixture of different Gaussian sources - is often hard to justify. It is therefore important to have additional information, which can steer the GMM estimation in the "right" direction. For example we may have access to the labels of *part* of the data set. Now the estimation problem belongs to the semi-supervised learning domain, since the estimation relies on both labeled and unlabeled points.

In this paper we focus on another type of side-information, in which *equivalence constraints* between a few of the data points are provided. More specifically, we use an unlabeled dataset augmented by *equivalence constraints* between pairs of data points, where the constraints determine whether each pair was generated by the same source or by different sources. We denote the former case as 'positive' constraints, and the latter case as 'negative' constraints, and present a method to incorporate them into an EM procedure.

What do we expect to gain from the semi-supervised approach to GMM estimation? We may hope that introducing side-information into the EM algorithm will result in faster convergence to a solution of higher likelihood. But much more importantly, our equivalence constraints should change the GMM likelihood function. As a result, the estimation procedure may choose *different* solutions, which would have otherwise been rejected due to their low relative likelihood in the unconstrained GMM density model. Ideally the solution obtained with side information will be more faithful to the desired results. A simple example demonstrating this point is shown in Fig. 1.

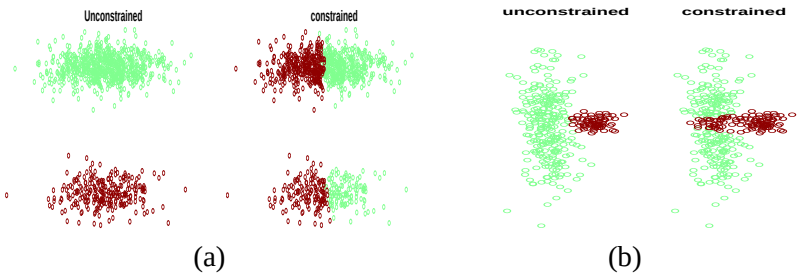

Figure 1: Illustrative examples to demonstrate the added value of *equivalence constraints*. (a) The data set consists of two *vertically aligned* classes: left - given no additional information, the EM algorithm identifies two *horizontal* classes, and this can be shown to be the maximum likelihood solution (with log likelihood of -3500 vs. log likelihood of -2800 for the solution shown on the right); right - additional side information in the form of equivalence constraints changes the probability function and we get a vertical partition as the most likely solution. (b) The dataset consists of two classes with partial overlap: left - without constraints the most likely solution includes two *non*-overlapping sources; right - with constraints the correct model with overlapping classes was retrieved as the most likely solution. In all plots only the class assignment of novel *un-constrained* points is shown.

Why do we use equivalence constraints, rather than partial labels as in prior work (summarized below)? Our basic observation is that unlike labels, in many unsupervised learning tasks equivalence constraints may be extracted with minimal effort or even automatically. One example is when the data is inherently sequential and can be modelled by a Markovian process. Consider for example a security camera application, where the objective is to find all the frames in which the same intruder appears. Due to the continuous nature of the data, intruders extracted from successive frames in rthe same clip can be assumed to come from the same person, thus forming positive constraints. In addition, two intruders which appear simultaneously in front of two cameras can not be the same person, hence a negative constraint is automatically established. Another analogous example is speaker segmentation and recognition, in which the conversation between several speakers needs to be segmented and clustered according to speaker identity. Here, it may be possible to automatically identify small segments of speech which are likely to contain data points from a single yet *unknown* speaker.

Knowing what to do with equivalence constraints comes in handy in some supervised learning settings as well. In what we call the 'distributed learning' scenario, we only have access to many independent and unccordinated teachers, each of whom see only a small fraction of the data.[1] For example, consider a database collected from locally labeled sets of images (with some overlap) from around the world. In the absence of coordination, the labels provided by the different teachers are inconsistent. Coordinating the labels of the different teachers can be almost as hard as labeling the original dataset. However, equivalence constraints can be easily extracted from the data provided by each teacher, and no further coordination is required.

Most of the work in the field of semi-supervised learning focused on the case of partial labels augmenting a large unlabeled data set [2, 3, 4]. A few recent papers use *equivalence constraints*, but without using any unlabeled data [5, 6, 7]. Two recent semi-supervised methods [8, 9] use both equivalence constraints and unlabeled data: In [8] *equivalence constraints* were introduced into the K-means clustering algorithm; this algorithm allows for the incorporation of both positive and negative constraints. In [9] equivalence constraints were introduced into the complete linkage clustering algorithm.

We describe comparative results in Section 3 using a number of data sets from the UCI repository and a large database of facial images [10]. Our experiments show that our algorithm gives significantly better clustering results, when compared with the two related algorithms mentioned above. One reason may be that the Gaussian mixture model is much more powerful as a clustering algorithm. More importantly, the probabilistic semantics of the EM procedure allows for the introduction of constraints in a principled way, thus overcoming many drawbacks of the heuristic approaches.

The rest of the paper is organized as follows: Section 2 presents our method of introducing equivalence constraints into EM. As it turns out, positive constraints can be easily incorporated into EM, while negative constraints require heavy duty inference machinery such as Markov networks. Hence we present the case of positive constraints and the case of negative constraints separately, and then discuss the case in which both types of constraints are provided. Experimental results are described in Section 3.

2 Constrained EM: the update rules

A Gaussian mixture model (GMM) is a parametric statistical model which assumes that the data originates from a weighted sum of several Gaussian sources. More formally, a GMM is given by: $p(x|\Theta) = \Sigma_{l=1}^{M} \alpha_l p(x|\theta_l)$ where M denotes the number of Gaussian sources in the GMM, α_l denotes the weight of each Gaussian, and θ_l denotes its respective parameters (μ_l its center and σ_l its covariance matrix).

EM is often the method of choice for estimating the parameter set of the model (Θ) using unlabeled data [1]. The algorithm iterates between two steps:

- 'E' step: calculate the expectation of the log-likelihood over all possible assignments of data points to sources.
- 'M' step: maximize the expectation by differentiating w.r.t the current parameters.

Equivalence constraints modify the 'E' step in the following way: instead of summing over *all* possible assignments of data points to sources, we sum only over assignments which comply with the given constraints. For example, if points x_i and x_j form a positive constraint, we only consider assignments in which both points are assigned to the *same* Gaussian source. On the other hand, if these points form a negative

[1] A related scenario (which we call 'generalized relevance feedback'), where users of a retrieval engine are asked to annotate the retrieved set of data points, has similar properties.

constraint, we only consider assignments in which each of the points is assigned to a *different* Gaussian source.

It is important to note that there is a basic difference between positive and negative constraints: While positive constraints are transitive (i.e. a group of pairwise positive constraints can be merged using transitive closure), negative constraints are not transitive. The outcome of this difference is expressed in the complexity of incorporating each type of constraint into the EM formulation. Therefore, we begin by presenting a formulation for positive constraints (Section 2.1), and then move on to negative constraints (Section 2.2). We conclude by presenting a unified formulation for both types of constraints (Section 2.3).

The following notations are used throughout:

- $p(x) = \sum_{l=1}^{M} \alpha_l \, p(x|\theta_l)$ denotes our GMM: each $p(x|\theta_l)$ is a Gaussian parametrized by $\theta_l = (\mu_l, \Sigma_l)$, with the mixing coefficient α_l, where $\sum_{l=1}^{M} \alpha_l = 1$.
- $\mathbf{X}$ denotes the set of all points, $\mathbf{X} = \{x_i\}_{i=1}^{N}$.
- $\mathbf{Y}$ denotes the assignment of all points to sources.
- E_Ω denotes the event $\{\mathbf{Y}$ complies with the constraints$\}$.

2.1 Incorporating positive constraints

In this setting we are given a set of unlabeled data points and a set of positive constraints. Since positive constraints may be grouped using transitive closure, we obtain (small) subsets of constrained points that share the same source. We call each subset a *chunklet*. Hence the data set is initially partitioned into chunklets, while unconstrained points form chunklets of size one. Let

- $\{X_j\}_{j=1}^{L}$ denote the set of all chunklets, and $\{Y_j\}_{j=1}^{L}$ denote the set of assignments of chunklet points to sources.
- The points which belong to a certain chunklet are denoted $X_j = \{x_j^1, \ldots, x_j^{|X_j|}\}$, where $\mathbf{X} = \bigcup_j X_j$.

In order to write down the likelihood of a given assignment of points to sources, a probabilistic model of how chunklets are obtained must be specified. We consider two such models:

1. Chunklets are sampled i.i.d, with respect to the weight of their corresponding source (points within each chunklet are also sampled i.i.d).
2. Data points are sampled i.i.d, without any knowledge about their class membership, and only afterward chunklets are selected from these points.

The first assumption is justified when chunklets are automatically obtained from sequential data with the Markovian property (see discussion in the introduction). The second sampling assumption is justified when *equivalence constraints* are obtained during *distributed learning*. When incorporating these sampling assumptions into the EM algorithm, different algorithms emerge: With the first assumption we obtain closed-form update rules for all of the GMM parameters. When the second sampling assumption is used there is no closed-form solution for the sources' weights. We therefore derive the update rules under the first sampling assumption, and then briefly discuss the second sampling assumption.

2.1.1 Deriving the update equations when chunklets are sampled i.i.d.

In order to derive the update equations of our Constrained GMM model, we must compute the expectation of the log likelihood, which is defined as:

$$E[log(p(\mathbf{X}, \mathbf{Y}|\Theta^{new}, E_\Omega))|\mathbf{X}\,\Theta^{old}, E_\Omega] = \sum_{\mathbf{Y}} log(p(\mathbf{X}, \mathbf{Y}|\Theta^{new}, E_\Omega)) \cdot p(\mathbf{Y}|\mathbf{X}, \Theta^{old}, E_\Omega) \tag{1}$$

In (1) $\sum_{\mathbf{Y}}$ denotes the summation over all assignments of points to sources: $\sum_{\mathbf{Y}} \equiv \sum_{y_1=1}^{M} \dots \sum_{y_N=1}^{M}$. In the following discussion we shall also reorder the sum according to chunklets: $\sum_{\mathbf{Y}} \equiv \sum_{Y_1} \dots \sum_{Y_L}$, where $\sum_{Y_j}$ stands for $\sum_{y_1^j} \cdots \sum_{y_{|X_j|}^j}$.

Calculating the Posterior probability Using Bayes rule and the assumption of chunklet independence, we can write

$$p(\mathbf{Y}|\mathbf{X}, \Theta^{old}, E_\Omega) = \frac{p(E_\Omega|\mathbf{Y}, \mathbf{X}, \Theta^{old})\, p(\mathbf{Y}|\mathbf{X}, \Theta^{old})}{\sum_{\mathbf{Y}} p(E_\Omega|\mathbf{Y}, \mathbf{X}, \Theta^{old})\, p(\mathbf{Y}|\mathbf{X}, \Theta^{old})} \tag{2}$$

From the by the definition of E_Ω it follows that

$$p(E_\Omega|\mathbf{Y}, \mathbf{X}, \Theta^{old}) = \prod_{j=1}^{L} \delta_{Y_j}$$

where $\delta_{Y_j} \equiv \delta_{y_1^j, \dots, y_{|X_j|}^j}$ equals 1 if all the points in chunklet i have the same source, and 0 otherwise.

Using the assumption of chunklet independence we have:

$$p(\mathbf{Y}|\mathbf{X}, \Theta^{old}) = \prod_{j=1}^{L} p(Y_j|X_j, \Theta^{old})$$

Therefore (2) can be rewritten as:

$$p(\mathbf{Y}|\mathbf{X}, \Theta^{old}, E_\Omega) = \frac{\prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|X_j, \Theta^{old})}{\sum_{Y_1} \cdots \sum_{Y_L} \prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|X_j, \Theta^{old})} \tag{3}$$

Computing the complete data likelihood The complete data likelihood can be written as:

$$p(\mathbf{X}, \mathbf{Y}|\Theta^{new}, E_\Omega) = p(\mathbf{Y}|\Theta^{new}, E_\Omega)\, p(\mathbf{X}|\mathbf{Y}, \Theta^{new}, E_\Omega) = p(\mathbf{Y}|\Theta^{new}, E_\Omega) \prod_{i=1}^{N} p(x_i|y_i, \Theta^{new})$$

where the last equality is due to the independence of datapoints, given the assignments to sources. Using Bayes rule and the assumption of chunklet independence, we can write:

$$p(\mathbf{Y}|\Theta^{new}, E_\Omega) = \frac{\prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|\Theta^{new})}{\sum_{Y_1} \cdots \sum_{Y_L} \prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|\Theta^{new})}$$

Let us use the notation $Z \equiv \sum_{Y_1} \ldots \sum_{Y_L} \prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|\Theta^{new})$. The likelihood can now be rewritten as:

$$p(\mathbf{X}, \mathbf{Y}|\Theta, E_\Omega) = \frac{1}{Z} \prod_{j=1}^{L} \delta_{Y_j}\, p(Y_j|\Theta^{new}) \prod_{i=1}^{N} p(x_i|y_i, \Theta) \tag{4}$$

Under the first sampling assumption introduced above, $\delta_{Y_j}\, p(Y_j|\Theta^{new}) = \alpha_{y_j}$. Hence $P(Y|\Theta^{new}, E_\Omega) = \prod_{j=1}^{L} \alpha_{y_j}$. It can also be easily shown that under this sampling assumption Z, the normalizing constant, equals 1. Therefore, the resulting log likelihood is

$$log\, p(\mathbf{X}, \mathbf{Y}|\Theta^{new}, E_\Omega) = \sum_{j=1}^{L} \sum_{x_i \in X_j} log\, p(x_i|y_i, \Theta^{new}) + \sum_{j=1}^{L} log(\alpha_{y_j})$$

Computing the expectation of the log likelihood We substitute (3) and (4) into (1) to obtain (after some manipulations) the following expression:

$$\begin{aligned} E(LogLikelihood) = \quad & \sum_{l=1}^{M} \sum_{j=1}^{L} \sum_{x_i \in X_j} log\, p(x_i|l, \Theta^{new}) \cdot p(Y_j = l|X_j, \Theta^{old}) \\ & + \sum_{l=1}^{M} \sum_{j=1}^{L} log\, \alpha_l \cdot p(Y_j = l|X_j, \Theta^{old}) \end{aligned} \tag{5}$$

where the chunklet posterior probability is:

$$p(Y_j = l|X_j, \Theta^{old}) = \frac{\alpha_l^{old} \prod_{x_i \in X_j} p(x_i|y_i^j = l, \Theta^{old})}{\sum_{m=1}^{M} \alpha_m^{old} \prod_{x_i \in X_j} p(x_i|y_i^j = m, \Theta^{old})}$$

In order to find the update rule for each parameter, we differentiate (5) with respect to μ_l, Σ_l and α_l, to get the following update equations:

$$\begin{aligned} \alpha_l^{new} &= \frac{1}{L} \sum_{j=1}^{L} p(Y_j = l|X_j, \Theta^{old}) \\ \mu_l^{new} &= \frac{\sum_{j=1}^{L} \bar{X}_j p(Y_j = l|X_j, \Theta^{old})|X_j|}{\sum_{j=1}^{L} p(Y_j = l|X_j, \Theta^{old})|X_j|} \\ \Sigma_l^{new} &= \frac{\sum_{j=1}^{L} \Sigma_{jl}^{new} p(Y_j = l|X_j, \Theta^{old})|X_j|}{\sum_{j=1}^{L} p(Y_j = l|X_j, \Theta^{old})|X_j|} \\ & \text{for} \quad \Sigma_{jl}^{new} = \frac{\sum_{x_i \in X_j} (x_i - \mu_l^{new})(x_i - \mu_l^{new})^T}{|X_j|} \end{aligned}$$

Above $\bar{X}_j$ denotes the sample mean of the points in chunklet j, $|X_j|$ denotes the number of points in chunklet j, and Σ_{jl}^{new} denotes the sample covariance matrix of the jth chunklet of the lth class.

As can be readily seen, the update rules above effectively treat each chunklet as a single data point weighted according to the number of elements in it.

2.1.2 Deriving the update equations when constraints are sampled i.i.d.

We now derive the update equations under the assumption that the data points are sampled i.i.d, and that chunklets are selected only afterwards. The difference between the two sampling assumptions first appears in the derivation folowing (4) above, since now the prior probabilities $p(Y_j|\Theta^{new})$ equal $\alpha_{y_j}^{|X_j|}$. We therefore have:

$$p(\mathbf{Y}|\Theta^{new}, E_\Omega) = \frac{\prod_{j=1}^{L} \alpha_{y_j}^{|X_j|}}{\prod_{j=1}^{L} \sum_{m=1}^{M} \alpha_m^{|X_j|}} \tag{6}$$

and the expected log likelihood becomes:

$$\sum_{l=1}^{M}\sum_{j=1}^{L}\sum_{x_i \in X_j} log\ p(x_i|l,\Theta^{new}) \cdot p(Y_j = l|X_j,\Theta^{old}) + \sum_{l=1}^{M}\sum_{j=1}^{L} |X_j| log\ \alpha_l \cdot p(Y_j = l|X_j,\Theta^{old})$$
$$- \sum_{j=1}^{L} log(\sum_{m=1}^{M} \alpha_m^{|X_j|}) \tag{7}$$

The difference between (5) and (7) lies in the last term, which can be interpreted as a "normalization" term. Differentiating (7) with respect to μ_l and Σ_l readily provides the same update equations as before, but now the posterior takes a slightly different form:

$$p(Y_j = l|X_j,\Theta^{old}) = \frac{(\alpha_l^{old})^{|X_j|} \prod_{x_i \in X_j} p(x_i|y_i^j = l,\Theta^{old})}{\sum_{m=1}^{M} (\alpha_m^{old})^{|X_j|} \prod_{x_i \in X_j} p(x_i|y_i^j = m,\Theta^{old})}$$

A problem arises with the derivation of update equations for the sources' weights α_l. In order to calculate α_l^{new}, we need to differentiate (7) subject to the constraint $\sum_{l=1}^{M} \alpha_l = 1$. Due to the "normalization" term we cannot obtain a closed-form solution, and we must resort to using a Generalized EM (GEM) scheme where the maximum is found numerically.

2.2 Incorporating negative constraints

As mentioned above incorporating negative constraints is inherently different and much more complicated than incorporating positive constraints. This difficulty can be related to the fact that unlike positive constraints, negative constraints are not transitive. For example if points x_i and x_j are known to belong to *different* classes, and points x_j and x_k are also known to belong to *different* classes, points x_i and x_k may or may not belong to the same class. Hence negative constraints are given as a group $\Omega = \{(a_i^1, a_i^2)\}_{i=1}^{P}$ of index pairs corresponding to P negatively constrained pairs.

Similar to the case of positive constraints, it is straightforward to write down the complete data likelihood.

$$p(\mathbf{X},\mathbf{Y}|\Theta, E_\Omega) = \frac{1}{Z} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \prod_{i=1}^{N} p(y_i|\Theta) p(x_i|y_i,\Theta) \tag{8}$$

Notice the similarity between (4) and (8), where the product over δ in (4) is replaced by the product over $(1-\delta)$ in (8). Also, the normalizing constant is now given by

$$Z \equiv \sum_{y_1} \dots \sum_{y_N} \prod_{\Omega} (1 - \delta_{y_{a_j^1}, a_j^2}) \prod_{i=1}^{N} p(y_i|\Theta).$$

In the following derivations we start with the update rules of μ_l and Σ_l, and then discuss how to update α_l, which once again poses additional difficulties.

Deriving the update equations for μ_l and Σ_l

Following exactly the same derivation as in the case of positive constraints, we can write down the update equations of μ_l and Σ_l:

$$\mu_l^{new} = \frac{\sum_{i=1}^N x_i p(y_i = l|\mathbf{X}, \Theta^{old}, E_\Omega)}{\sum_{i=1}^N p(y_i = l|\mathbf{X}, \Theta^{old}, E_\Omega)} \qquad \Sigma_l^{new} = \frac{\sum_{i=1}^N \widehat{\Sigma_i} l p(y_i = l|\mathbf{X}, \Theta^{old}, E_\Omega)}{\sum_{i=1}^N p(y_i = l|\mathbf{X}, \Theta^{old}, E_\Omega)}$$

where $\widehat{\Sigma_i} l = (x_i - \mu_l^{new})(x_i - \mu_l^{new})^T$ denotes the sample covariance matrix.

The difficulty lies in calculating the posterior probabilities $p(y_i = l|\mathbf{X}, \Theta^{old}, E_\Omega)$, which are calculated by marginalizing the following expression:

$$p(\mathbf{Y}|\mathbf{X}, \Theta^{old}, E_\Omega) = \frac{\prod_{(a_i^1, a_i^2)}(1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \prod_{i=1}^N p(y_i|x_i, \Theta^{old})}{\sum_{y_1} \cdots \sum_{y_N} \prod_{(a_i^1, a_i^2)}(1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \prod_{i=1}^N p(y_i|x_i, \Theta^{old})} \tag{9}$$

It is not feasible to write down an explicit derivation of this expression even for a very small number of constraints, since the probability of a certain assignment of point x_i to source l depends on the assignments of all other points to which x_i is negatively constrained. However, since the dependencies enforced by the constraints are local, we can describe (8) as a product of local components, and therefore it can be readily described using a Markov network.

A Markov network is a graphical model defined by a graph $G = (V, E)$, whose nodes $v \in V$ represent a random variable and whose edges E represent the dependencies between the different nodes. In our case, a data point x_i is represented by two nodes in the graph: an observable node o_i and a hidden node h_i. The hidden node h_i describes its source label, while the data point itself o_i is an observed example from the source (see Fig. 2). Each observable node o_i is connected to its hidden node h_i by a directed edge, holding the potential $p(x_i|y_i, \Theta)$. Each hidden node h_i also has a local potential in the form of $p(y_i|\Theta)$, reflecting the prior on the source weights. A negative constraint between data points x_i and x_j is represented by an *undirected* edge between their corresponding hidden nodes h_i and h_j, having a potential of $(1 - \delta_{h_i, h_j})$. Intuitively these edges prevent both hidden variables from having the same value.

The mapping of our problem into the language of graphical models makes it possible to use efficient inference algorithms. We use Pearl's junction tree algorithm [11] to compute the posterior probabilities. The complexity of the junction tree algorithm is exponential in the induced-width of the graph, hence for practical considerations the number of negative constraints should be limited to $O(N)$.[2] Therefore, in order to achieve scalability to large sets of constraints, we must resort to approximations; in our implementation we specifically replaced the graph by its spanning tree.

Deriving the update equations for α_l

The derivation of the update rule of $\alpha_l = p(y_i = l|\Theta_{new}, E_\Omega)$ is more intricate due to the normalization constant Z. In order to understand the difficulties, note that maximizing the expected log-likelihood with

[2]The general case with $O(N^2)$ constraints is NP-hard, as the graph coloring problem can be reduced to it.

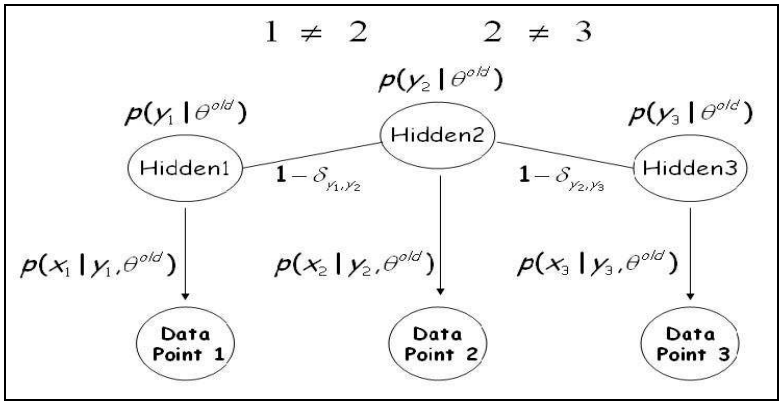

Figure 2: An illustration of the Markov network required for incorporating negative constraints. Data points 1 and 2 have a negative constraint, and so do points 2 and 3.

respect to α_l is equivalent to maximizing:

$$\mathcal{I} = -log(Z) + \sum_{m=1}^{M}[\sum_{i=1}^{N} p(y_i = m|X, \Theta, E_\Omega)]log(\alpha_m)$$

where the normalization factor Z is:

$$Z = p(E_\Omega|\Theta) = \sum_{\mathbf{Y}} p(\mathbf{Y}|\Theta)p(E_\Omega|\mathbf{Y}) = \sum_{y_1} ... \sum_{y_N} \prod_{i=1}^{N} \alpha_{y_i} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \tag{10}$$

The gradient of this expression w.r.t. α_l is given by

$$\frac{\partial \mathcal{I}}{\partial \alpha_l} = -\frac{1}{Z}\frac{\partial Z}{\partial \alpha_l} + \frac{\sum_{i=1}^{N} p(y_i = l|X, \Theta, E_\Omega)}{\alpha_l} \tag{11}$$

Equating (11) to 0 (subject to the constraint $\sum_{l=1}^{M} \alpha_l = 1$) does not have a closed form solution, and once again we must use the numerical GEM procedure. The new difficulty, however, lies in estimating (11) itself; although the posterior probabilities have already been estimated using the Markov network, we still need to calculate Z and its derivatives. This calculation is bound to be difficult, as suggested by the similarity between Z and $\frac{\partial Z}{\partial \alpha_l}$ and the posterior probabilities.

To address this new difficulty, we considered two different approaches for calculating Z and its derivatives[3]. In the first approach we perform an exact calculation of both terms using additional Markov networks, and in the second approach we use an approximation based on a pseudo-likelihood assumption.

Calculating Z and $\frac{\partial Z}{\partial \alpha_l}$ exactly: When comparing (9) and (10), it is evident that Z can be calculated using a Markov network. This network has a similar structure to the former network: it contains the same hidden nodes and local potentials, but lacks the observable nodes (see Fig 3). Calculating Z then amounts to an elimination of all the variables in this network.

[3]As the simplest brute-force alternative we may completely ignore Z and its derivatives, which leads to a closed form update rule for α_l. However, we have observed that this solution tends to degrade the algorithm's performance.

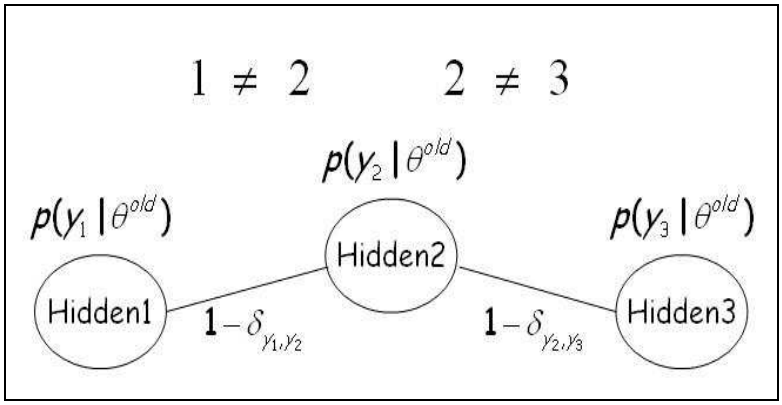

Figure 3: An illustration of the Markov network required for calculating Z, for the case where data points 1 and 2 have a negative constraint, as do points 2 and 3.

As for $\frac{\partial Z}{\partial \alpha_l}$, each of the M derivatives requires its own Markov network. The derivatives are given by:

$$\frac{\partial Z}{\partial \alpha_l} = \sum_{y_1} ... \sum_{y_N} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \frac{\partial}{\partial \alpha_l} \prod_{i=1}^{N} \alpha_{y_i} = \sum_{y_1} ... \sum_{y_N} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \sum_{j=1}^{N} \prod_{i=1, i \neq j}^{N} \alpha_{y_i}$$

and the value of each derivative is calculated by eliminating all the variables, just as for Z.

Computing Z and its gradients is equivalent to $M + 1$ elimination processes, whose complexity is exponential in the induced-width of the graph. Since the gradient computation is performed many times in each EM round, this method can be rather slow for complicated constraint graphs.

Approximating Z using the pseudo-likelihood assumption: Z can be approximated under the assumption that the negative constraints are mutually exclusive. Denote the number of negative constraints by c. If we now assume that all pairs of constrained points are disjoint, the number of unconstrained points is $u = N - 2c$. Assume, without loss of generality, that the unconstrained data points are indexed by $1 \ldots u$, and the remaining points are ordered so constrained points are given successive indices (points $u + 1$ and $u + 2$ are negatively constrained, etc.). Hence Z can be decomposed as follows:

$$\begin{aligned} Z &= \sum_{y_1} ... \sum_{y_N} \prod_{i=1}^{N} \alpha_{y_i} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \\ &= \sum_{y_1} \alpha_{y_1} ... \sum_{y_u} \alpha_{y_u} \sum_{y_{u+1}} \sum_{y_{u+2}} \alpha_{y_{u+1}} \alpha_{y_{u+2}} (1 - \delta_{y_{u+1}, y_{u+2}}) ... \sum_{y_{N-1}} \sum_{y_N} \alpha_{y_{N-1}} \alpha_{y_N} (1 - \delta_{y_{N-1}, y_N}) \quad (12) \\ &= (1 - \sum_{i=1}^{M} \alpha_i^2)^c \end{aligned}$$

This expression for Z may be easily differentiated, and can be used in a GEM scheme. Although the assumption is not valid in most cases, it seems to yield a good approximation for sparse networks. We empirically compared the three approaches presented. As can be expected, the results show a trade-off between speed and accuracy. However, the average accuracy loss caused by ignoring or approximating Z seems to be small.

2.3 Combining positive and negative constraints

Both types of constraints can be incorporated into the EM algorithm using a single Markov network by a rather simple extension of the network described in the previous section. Assume we have, in addition to the negative constraints, a set $\{c_i\}$ of chunklets, where each c_i is a list of points' indices, known to share the same label[4]. The likelihood becomes

$$p(X, Y|\Theta, E_\Omega) = \frac{1}{Z} \prod_{c_i} \delta_{y_{c_i}} \prod_{(a_i^1, a_i^2)} (1 - \delta_{y_{a_i^1}, y_{a_i^2}}) \prod_{i=1}^{N} p(y_i|\Theta) p(x_i|y_i, \Theta)$$

where $\delta_{y_{c_i}}$ is 1 iff all the points in chunklet c_i have the same label. Since the probability is non-zero only when the hidden variables in the chunklet are identical, we can replace the hidden variables of each chunklet $h_{i_1} \cdots h_{i_{|c_i|}}$ with a single hidden variable. Hence in the Markov network implementation positively constrained points share a hidden father node (see Fig 4). The EM procedure derived from this distribution is similar to the one presented above, requiring only a modified normalizing constant Z.

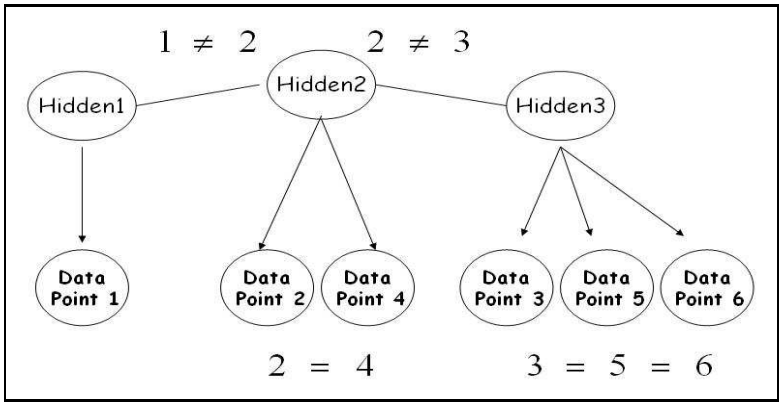

Figure 4: An illustration of the Markov network required for incorporating both negative and positive constraints. Data points 1 and 2 have a negative constraint, and so do points 2 and 3. Data points 2 and 4 have a positive constraint, and so do points 3,5 and 6.

3 Experimental results

In order to evaluate the performance of our EM derivations and compare it to the performance of the constrained K-means algorithm presented in [8], we tested our algorithms using several data sets from the UCI repository. We simulated a 'distributed learning' scenario in order to obtain side information. In this scenario, we obtain *equivalence constraints* using the help of N teachers. Each teacher is given a random selection of K data points from the data set, and is then asked to partition this set of points into equivalence classes. The constraints provided by the teachers are gathered and used as *equivalence constraints*. We compared the performance of the following algorithms:

- K-means algorithm when no side information is used.

[4] In this section , positive constraints are sampled in accordance with the first sampling assumption described in Section 2.1, as the data points are assumed to be i.i.d before the introduction of the constraints.

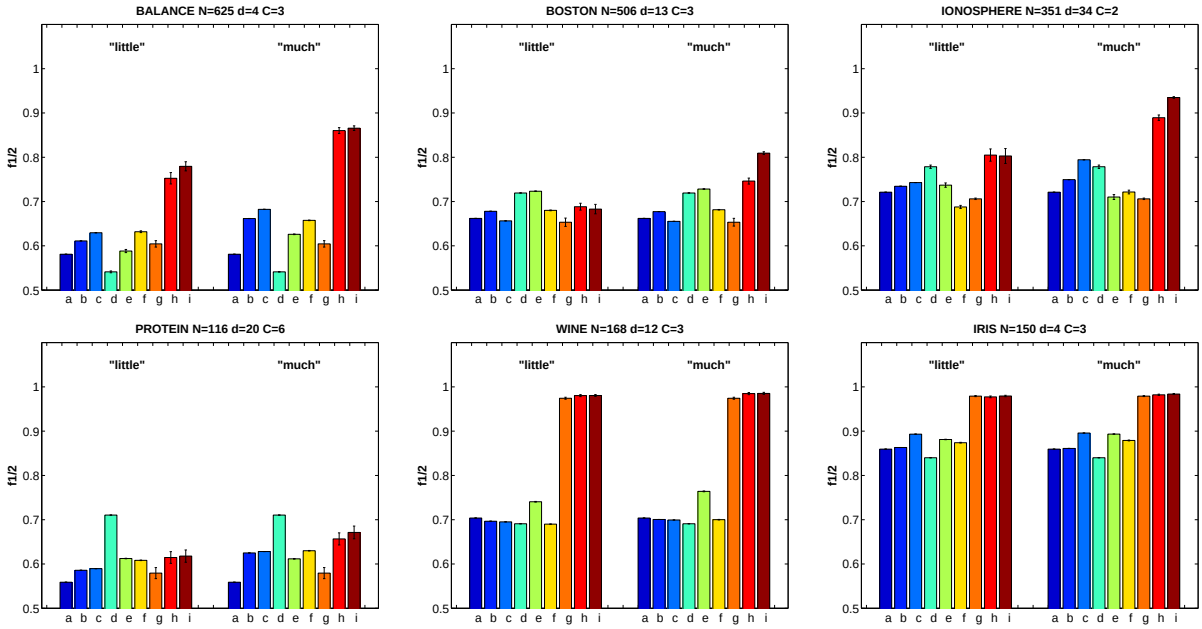

Figure 5: Combined precision and recall scores ($f_{\frac{1}{2}}$) of several clustering algorithms over 5 data sets from the UCI repository. Results are presented for the following algorithms: (a) K-means, (b) constrained K-means using only positive constraints, (c) constrained K-means using both positive and negative constraints, (d) regular EM, (e) EM using positive constraints, and (f) EM using both positive and negative constraints. Results are shown twice, using 15% of the data points in constraints (left bars) and 30% of the points constrained (right bars). The results were averaged over 100 realizations of constraints. Also shown are the names of the data sets used and some of their parameters: N - the size of the data set; C - the number of classes; d - the dimensionality of the data.

- Constrained K-means [8], using only positive *equivalence constraints*.
- Constrained K-means [8], using both positive and negative *equivalence constraints*.
- EM of a Gaussian mixture model when no side information is used.
- Our closed form EM algorithm when only positive *equivalence constraints* are used.
- Our Markov network EM algorithm, using both positive and negative *equivalence constraints*.

The number of constrained points was determined by the number of teachers N and the size of the subset K that they were each given. By controlling the product NK we modified the amount of side information provided to the different algorithms. Specifically, we experimented with two conditions: using "little" side information (approximately 15% of the data points are constrained) and using "much" side information (approximately 30% of the points are constrained).[5] All algorithms were given the same initial conditions that did not take into account the available *equivalence constraints*. The clustering obtained was evaluated using a combined measure of precision P and recall R scores: $f_{\frac{1}{2}} = \frac{2PR}{R+P}$

The results over several UCI data sets are presented in Fig. 5. Several effects can be clearly seen:

[5] **TODO** on two of the datasets presented we used more side-information, since the amounts described above showed little or no improvement.

- As expected, our constrained EM algorithm outperforms the constrained K-means algorithms on all databases, and shows substantial improvement over the baseline EM as well.
- Introducing side information in the form of *equivalence constraints* clearly improves the results of both K-means and the EM algorithms. As the amount of side-information increases, the algorithms which make use of it tend to improve.
- Most of the improvement can be attributed to the positive constraints, and can be achieved using our closed form EM version. In most cases adding the negative constraints contributes a small but significant improvement over results obtained when using only positive constraints.

Figure 6: Three images of the same person from the YaleB data set.

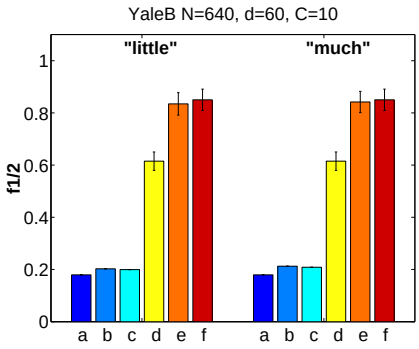

Figure 7: combined precision and recall scores of several clustering algorithms over the YaleB facial data set. The results are presented using the same format as in Fig. 5, representing an average over more than 1000 realizations of constraints. Percentage of data in constraints was 50% (left bars) and 75% (right bars). It should be noted that when using 75% of the data in constraints, the constrained K-means algorithm failed to converge in more than half of its runs.

It should be noted that most of the UCI data sets considered so far contain only two or three classes. Thus in the 'distributed learning' setting a relatively large fraction of the constraints were positive. In a more realistic situation, with a large number of classes, we are likely to gather more negative constraints than positive constraints. This is an important point in light of the results in Fig. 5, where the major boost in performance was due to the use of positive constraints.

In order to see what happens with many classes, we conducted the same experiment using a subset of the YaleB facial image dataset [10] which contains a total of 640 images, including 64 frontal pose images of 10 different subjects. In this database the variability between images of the same person is due mainly to different lighting conditions. We automatically centered all the images using optical flow. Images were then converted to vectors, and each image was represented using the first 60 principal components coefficients. The task was to cluster the facial images belonging to these 10 subjects.

Some example images from the data set are shown in Fig. 6. Due to the random selection of images given to each of the N teachers, most of the constraints obtained were indeed negative. Our results are summarized in Fig. 7. We see that even though there were only a small number of positive constraints, most of the beneficial effect of constraints is obtained from looking at this small subset of positive constraints; as before, our constrained algorithms all substantially outperformed the regular EM algorithm.

4 Summary

We have shown how equivalence constraints can be incorporated into the EM algorithm, in order to compute in a semi-supervised manner a Gaussian Mixture Model of the data. When using only positive constraints, we provided an efficient closed form solution for the update rules, and demonstrated that using positive constraints can significantly boost clustering performance. When negative constraints are added, the computational cost increases since a Markov network is used as an inference tool, and we must defer to approximations of the source weights update rules. Our experiments show that most of the improvement in performance is obtained from the positive constraints alone, with some small (but significant) contribution from the negative constraints (which are also harder to use efficiently).

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSSB*, 39:1–38, 1977.

[2] D. Miller and S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *NIPS 9*, pages 571–578. MIT Press, 1997.

[3] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 14. The MIT Press, 2001.

[4] K. Nigam, A.K. McCallum, S. Thrun, and T.M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

[5] P.J. Phillips. Support vector machines applied to face recognition. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *NIPS 11*, page 803ff. MIT Press, 1998.

[6] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In M. Nielsen A. Heyden, G. Sparr and P. Johansen, editors, *Computer Vision - ECCV 2002*, volume 4, page 776ff, 2002.

[7] E.P Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learnign with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, volume 15. The MIT Press, 2002.

[8] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proc. 18th International Conf. on Machine Learning*, pages 577–584. Morgan Kaufmann, San Francisco, CA, 2001.

[9] D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering, 2002.

[10] A. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Generative models for recognition under variable pose and illumination. *IEEE international Conference on Automatic Face and Gesture Recognition*, pages 277–284, 2000.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.