Experimental Results on Manifold Expectation Maximization Clustering

Xinyuan Cao, Mark Dijkstra, Linyun He, Yuze Zhou

1. Introduction

The problem that we try to address is the clustering of manifolds. This is a worthwhile problem to study because it gives incentive to new unsupervised learning methods to learn underlying patterns in data that may have not been initially thought to have a manifold structure. However, it is also a question of how often data in nature inherently has such a structure. Yet, the development of these algorithms have application in video image processing, and as a result do have reasons in being studied. Our goal is to implement an algorithm that is somewhat unique in nature, but still hopefully accomplishes the goal of cluster multi-manifold data. There is the obvious concern that since this is clustering of data in an unsupervised setting, it becomes difficult to measure the accuracy of such a cluster. In order to circumvent this problem to measure the quality of our algorithm we construct our problem in a very specific manner. We begin with a dataset that is already partitioned into 4 clusters in 2-dimensions. Then we create our 3-dimensional manifold from this 2-dimensional space using a non linear transformation function with some Gaussian noise added along. Then we attempt to recreate the original clustering using our expectation maximization algorithm. We measure the effectiveness of our algorithm using the Purity measure. Our main contributions lie in the study of how to initialize the points for our Expectation Maximization algorithm that we had development. This is a meaningful endeavor, because we have noticed that initialization greatly affects the outcome of the clusters created. Thus, without a decent initialization strategy, an Expectation Maximization Algorithm will fail to cluster the points in a manner that is expected. This will be explored in our challenges subsection of our methods section. We then proceed to visually compare the results of our algorithm versus ones that already exist. We are also curious to see against what sorts of datasets our algorithm fails, and ideally analyze why they fail. In our algorithm we make the assumption that our manifolds are sampled from a geodesic Gaussian distribution. We also make the assumption that we know the number of clusters, k, and the dimensionality of the embedded space of the manifold, d.

2. Related Work

There has been work on creating expectation- maximization algorithms to cluster manifolds. Formally, this problem has been known as the multi-manifold learning problem. The classic variant of manifold learning expects a single manifold lying in a high dimensional space such that it has an intrinsic lower dimensional representation that is interesting to find. Algorithms such as ISOMAP achieve this goal well when applied to a single manifold. However, when the underlying data comes from multiple manifolds, or simply when there are large gaps between the manifolds, ISOMAP fails. The problem arises in the creation of the graph G consisting of the edges between the points sampled from the manifolds (because the k nearest neighbors method or the ϵ radius method fails to connect points separated by a large gap). Thus, there is an infinite distance between certain points from separated manifolds in the distance matrix D_G. In the paper [1] they solved this problem. Formally, given two points $i \in M_1, j \in M_2$ such that $M_1.M_2$ are two distinct manifolds (i.e. gap between them in the metric space R^D), the distance computed by the construction of the distance matrix D_G between i and j is ∞ (i.e. $d_G(i,j) = \infty$). The solution is to take two points $i' \in M_1, j' \in M_2$ that minimizes the euclidean distance between M_1 and M_2 and then computing $d_G(i,j) := d_G(i,i') + d_{R^D}(i',j') + d_G(j,j')$. After getting rid of the infinite distances in D_G, the rest of the augmented ISOMAP algorithm proceeds as usual. This following paper is more closely related to our goals at hand, *Manifold Clustering*, deals with the multi-manifold learning problem through the lens of the Expectation Maximization problem. Given a set of Manifolds (their terminology is 'images', because they work with an image dataset) $\{X_1, \ldots, X_n\} \in R^D$ and a desired set of k non intersecting non-linear manifolds that can be embedded in $d << D$ dimensions (i.e. we know the number of manifolds. and their intrinsic dimensionality already). They want to learn the labeling of the data (i.e. which cluster/manifold they belong to) and the parameters of the manifolds; this is done through a variant of an Expectation Maximization algorithm. This

EM algorithm begins with the construction of a geodesic matrix the same way as done in ISOMAP. Then a $k \times n$ weight matrix W, whose entries give the probability that X_i belongs to manifold c, is randomly initialized. The next two steps are repeated until convergence.

M-Step: For each class c, supply $\vec{w}_c$ and D and use node-weighted MDS to embed the points in d dimensions.
E-Step: For each point, estimate the distance to the manifolds implied by the **M-Step** and then reweight.

Furthermore, there has been work with initialization of cluster centers with respect to the clustering problem and expectation maximization problems. Our influence in our work stems from a paper that discusses rank-order distance based clustering.[2]. Although we do not implement their algorithm precisely, a lot of our work with our nearest neighbor merge algorithm is influenced by it. This paper explores the notion of assigning rank based on calculating how many neighborhoods a given document (or data point for all intents and purposes) is a contained in. From here, the paper assigns the documents that lie in more neighborhoods of other documents to be somewhat more important or significant. The points are then merged under a normalized distance metric defined in the literature (not relevant to our purposes). The goal of this algorithm is to do the clustering itself; however, our goal, as we will describe in subsection 3.4 is merely to initialize the cluster centers.

3. Methods

Our goal is to take in real world data and see at which point our algorithm fails. Our algorithm is based on a combination of the papers we have read and described in the previous section.

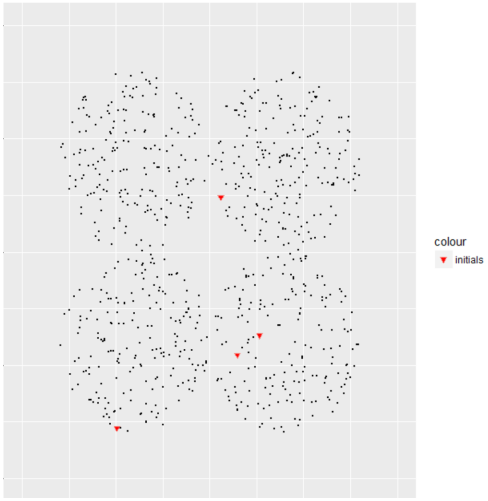
3.1 Challenges

The challenges we faced were similar to the ones that the authors of the papers that inspired our work faced. For example, it is the case that a standard expectation maximization algorithm, and ISOMAP, will fail if the graphs constructed during the initial stages of ISOMAP are separated by a wide gap. In other words; our graph is comprised of multiple connected components as opposed to a single one. The solution we chose to implement was to estimate the distance between two nodes i and j from different connected components C_i and C_j using the following formula [3]:

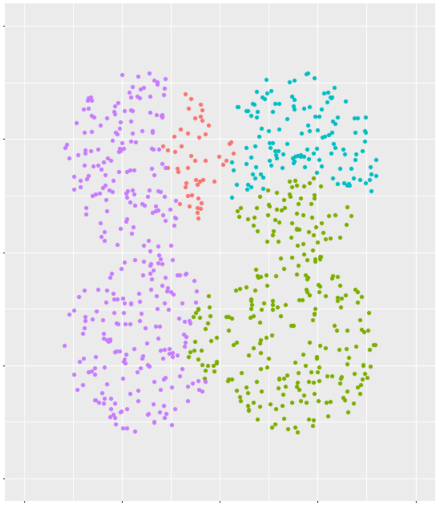
$$d_G(i,j) := d_G(i,i') + d_x(i',j') + d_G(j,j')$$

$$(i',j') = \arg \min_{(n,m):n \in C_i, m \in C_j} d_X(n,m)$$

where $d_G(\cdot,\cdot)$ denotes the geodesic distance and $d_G(\cdot,\cdot)$ denotes the Euclidean distance. This method would be further described in more detail in related works section of this paper. Another major challenge was accurately initializing the points for the expectation maximization portion of our algorithm. The initial points had a significant bearing on the quality of the clustering exhibited on our toy data. Typically, the best initialization required each of the initial points to lie somewhere on each of the k clusters we wished to see after running the algorithm. If we meticulously chose the starting points we had good results, this will be shown in the experiments section later on. We tried to combat this using a various number of methods, many which have failed. For example, we tried to use a $NN - rank$ method. The algorithm ordered the points that belonged to the most neighborhoods of other points in ascending order, and choose the top k points (such that none of these k points are in each others neighborhoods, thus choosing the next largest point to replace a point which is already in the neighborhood of a point already chosen) to represent the initial positions of the k clusters.



We notice that two of the red triangles seem to lie within a single one of the original cluster's data points (note that the points are currently colored in black, because the algorithm we will run does not know the original cluster that we imposed prior to the transformation). This will be a key factor in determining the result of our expectation maximization algorithm, and a motivation for why we decide to alter the initialization of our algorithm to somehow separate these initial points to belong somewhere within a predefined cluster we hope to recover. Thus, running our expectation maximization algorithm on this 2-D representation of our 3-D generated manifold data gave us an unfavorable clustering:



We managed to implement three node merge algorithms to initialize our starting centers for expectation maximization algorithm to give us more favorable results. The premise of all three of our node merge algorithms(nearest neighbors min merge, nearest neighbors max merge, nearest neighbors stochastic merge) is that we will compare a set of nodes and then through some decision process (depending on which of the three algorithms is selected) select k nodes to represent the k initial points for our expectation maximization algorithm. They will be described in further detail in section 3.4 and in the Experimental Results section.

3.2 Assumptions

Assume that the data is scattered across k different clusters on a set of manifolds, $\mathcal{M} := \{M_1, \ldots, M_2\}$ in R^D whose intrinsic dimension is in R^d. For each of cluster $M_i \in \mathcal{M}$, M_i is generated from a geodesic Gaussian distribution with the probability density function:

$$f_i(x) \propto \frac{1}{\sigma_i^d} e^{-\frac{d_G^2(x,\mu_i)}{2\sigma_i^2}}$$

Where μ_i is the mean of the Gaussian distribution, σ_i^2 is the variance of the Gaussian distribution, and $d_G(x, y)$ corresponds to the geodesic distance for manifold M_i.

3.3 Data-sets

We chose to operate on synthetic data to establish the validity of our algorithm. Since our algorithm's goal is to identify clusters which represent manifolds from a mixture of manifolds in some higher dimensional space, our synthetic data was created specifically to test this concept. Specifically, we took four clusters in R^2 which we then transformed into manifolds in R^3 and then attempted to run our algorithm on this transformed manifold data set to recover the original clustering in R^2 as accurately as possible.

Afterwards, we chose to operate on famous datasets such as MNIST and realized that our algorithm fails to work in this setting due to its high sparsity low rank data matrix on a high dimensional setting. We acknowledge that other algorithms used for similar problems are capable of running on MNIST and other datasets that lie on manifolds such as the COIL dataset, which is a dataset of pictures of images taken at many angles to.

3.4 Algorithm

Estimating Geodesic Distance Algorithm:

1. Determine neighbors of each point using k-Nearest Neighbors of ϵ-neighborhood

2. Construct a neighborhood graph (nearest neighbors get connected) where edge weight is euclidean distance

3. Compute shortest path distance between all pair of points and use it as the geodesic distance

4. if more than one connected components are detected, calculate heuristic geodesic distance by: $d_G(i,j) = d_G(i,i') + d_x(i',j') + d_G(j,j')$ as previously described

Our expectation maximization algorithm works by iterating between its Expectation and Maximization steps in order to find the best possible centers for our clusters. We acknowledged that arbitrarily setting the initial centers before running our EM algorithm would result in poor clustering (even for the simple toy data we create) as described in the previous section; hence, we implemented an algorithm to hedge the risk that our centers would be arbitrarily bad. We will now describe the three merge algorithms at a high level.

Center Initialization Using Merge Algorithm

Each merge algorithm initially computes a set of "important points" (such that the number of important points is $\geq k$, where k is the number of clusters we want our algorithm to return) from which we will compute a subset of k important points which will indicate the initial cluster centers for our expectation maximization algorithm. A point is considered to be important if it lies in the neighborhoods of more than s points, where s is a threshold passed into the merge algorithm as a parameter. Typically, we want $s \geq K$ where K is the number of neighbors used in our K nearest neighbors algorithm preprocessing step. It is important to note that although the preprocessing step is K nearest neighbors, some points will have more than K neighbors (hence why we choose s to be the threshold for our merge algorithm, where $s \geq K$). We compute K neighbors for each point independently. Thus, it can be the case that we compute the K neighbors for point x_i and then we check point x_j which is currently not in the neighborhood of x_i, we can see that x_i may be closer to x_j than K other points when computing shortest distances with respect to x_j, thus giving x_i a $K + 1^{th}$ neighbor and possibly so forth.

Recall, k is the number of clusters our algorithm assumes our dataset to have and S is the set of important points. Our three algorithms operate as so:

1. Stochastic min merge: given two points, x_1, x_2, (such that $x_1, x_2 = \underset{x_1,x_2 \in X}{argmin}(d(x_1, x_2))$), this merge algorithm stochastically removes one of the two points with equal probability. This process repeats until there are k points left.

2. Nearest neighbor min merge: given two points, x_1, x_2, (such that $x_1, x_2 = \underset{x_1,x_2 \in X}{argmin}(d(x_1, x_2))$) and two neighborhoods $N(x_1), N(x_2)$. To note that $N(x_1)$ and $N(x_2)$ are generated from all vertical set, not only from S. This algorithm computes the "connection" of each node to its neighbors by computing $\sum_{y \in N(x_1)} dist(x_1, y)$ and $\sum_{y \in N(x_2)} dist(x_2, y)$. Remove the one with bigger sum of distances from S and continue the procedure until k nodes left.

3. Nearest neighbor max merge: given two points, x_1, x_2, (such that $x_1, x_2 = \underset{x_1,x_2 \in X}{argmax}(d(x_1, x_2))$) and two neighborhoods $N(x_1), N(x_2)$. Compare the "connection" as the same in 2. Choose the one with smaller sum of distances and merge it with its nearest neighbors in S. Iterate until k nodes left in S.

Empirical results show that the min merge method is the most effective among them all, which will be further illustrated in the experiment part. The explicit procedure for which is as followed:

Algorithm 1 Min Merge Algorithm

procedure MIN MERGE
  $D_G \leftarrow$ Approximate Geodesic Distance
  $c \leftarrow$ The criterion for initial important nodes
  $k \leftarrow$ The number of initial nodes
  $C \leftarrow$ The set of nodes that are in the neighbourhood of more than k others
  while $|C| > k$ **do**
    Find $(u, v) \in C$ such that $d_G(u, v) = \min_{(u',v') \in C} d_G(u', v')$
    if $\sum_{x \in N(u)} d_G(u, x) \leq \sum_{x \in N(v)} d_G(v, x)$ **then**
      Delete v from C
    else
      Delete u from C

Expectation Maximization Algorithm on Manifolds Now that we have the k initial cluster centers that will be used in our expectation maximization algorithm, which goes as followed

Algorithm 2 Expectation Maximization on Manifolds

procedure MANIFOLD EM
  $D_G \leftarrow ApproximateGeodesicDistance$
  $C \leftarrow MergeInitilization$
  while not converged **do**
    for $j = 1, \ldots, n$ **do**
      $y_j = argmax_{1 \leq i \leq k} \frac{1}{(\sigma_i^2)^{\frac{d}{2}}} e^{-\frac{d_G^2(x_j, \mu_i)}{2\sigma_i^2}}$
    for $i = 1, \ldots, k$ **do**
      $\pi_i = \frac{|C_i|}{\sum_{j=1}^{k} |C_j|}$
      $\mu_i = argmin_{x \in C_i} \sum_{x_j \in C_i} d_G^2(x, x_j)$
      $\sigma_i^2 = \frac{\sum_{x_j \in C_i} d_G^2(x_j \mu_i)}{d|C_i|}$

The E-Step of the EM algorithm on manifolds is the same as that of the traditional one. In order to clarify the algorithm, we only have to look at the M-Step. First recall the likelihood function and log-likelihood function given n samples $x_1 \cdots x_n$ from a geodesic Gaussian distribution with mean μ and variance σ:

$$L(\mu, \sigma^2; x) \propto \prod_{i=1}^{n} \frac{1}{\sigma^d} e^{-\frac{d_G^2(x_i, \mu)}{2\sigma^2}}$$

$$l(\mu, \sigma^2; x) \propto \sum_{i=1}^{n} [-\frac{d_G^2(x_i, \mu)}{2\sigma^2} - log(\sigma^d)]$$

To maximize $l(\mu, \sigma; x)$ with respect to μ, we have to find $\hat{mu}$ on the manifold minimizing $\sum_{i=1}^{n} d_G^2(x_i, \mu)$. However, since the structure of the underlying manifold is unknown and the only thing we have are the data points on it, we estimate μ instead by the center of the data, that is:

$$\hat{\mu} = \arg\min_{x_j} \sum_{i=1}^{n} d_G^2(x_i, x_j)$$

Now that we already get an estimate for $\hat{\mu}$, we now move to finding the optimal $\hat{\sigma^2}$. First by taking the first-order derivatives of $l(\mu, \sigma^2; x)$ with respect to σ^2 and setting it to zero:

$$\frac{\partial l}{\partial \sigma^2} \propto \frac{\sum_{i=1}^{n} d_G^2(x_i, \mu)}{2\sigma^4} - \frac{nd}{2} \frac{1}{\sigma^2} = 0$$

Therefore the maximal likelihood estimate for σ^2 is:

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{n} d_G^2(x_i, \mu)}{nd}$$

Benchmark for Clustering Algorithms Our algorithm's accuracy is tested using the Purity Measure [4].

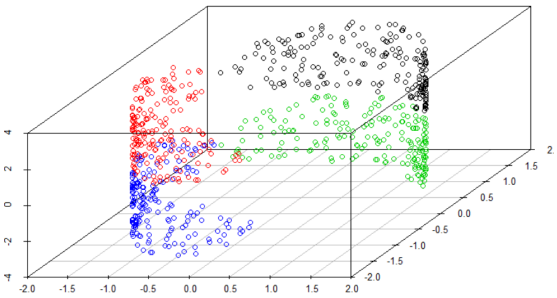
$$purity(\Omega, \mathcal{C}) = \frac{1}{N} \sum_{k} max_j |w_k \cap c_j|$$

1. Ω is the original set of clusters
2. $\mathcal{C}$ is the set of clusters produced by our algorithm
3. N is the number of total data points

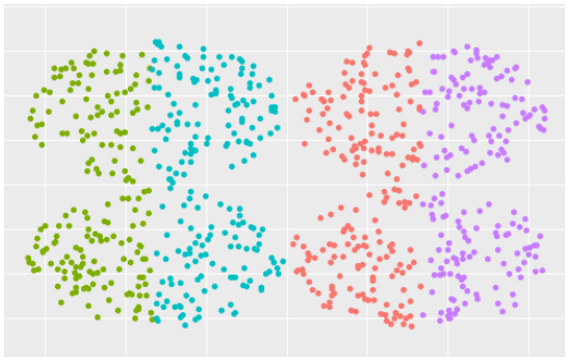
Empirically speaking, the larger the purity score is, the better clustering result is returned by the algorithm. If our algorithm has a higher purity score than the others, then it could much better discover the underlying clustering structures on the manifold than the traditional methods like EM and spectral clustering.

4. Experimental Results

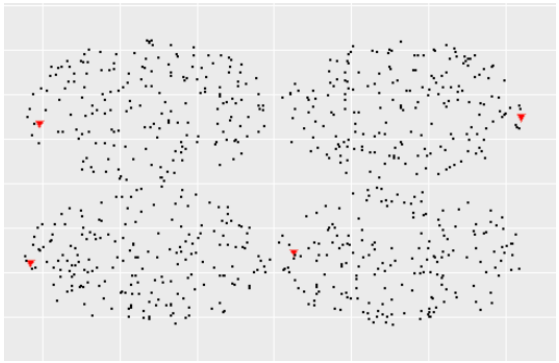
We started our algorithm experimentation on a generated data-set which looks like points sampled from a set of manifolds.



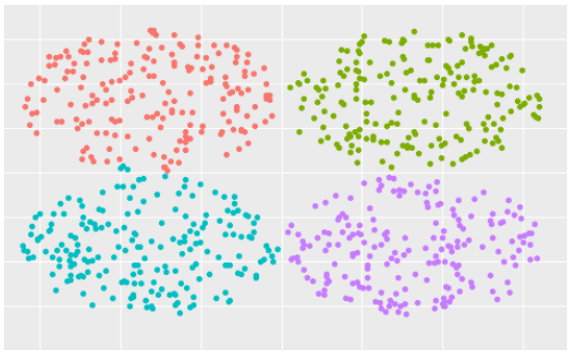
The different colors indicate the different manifolds used to construct the manifold. Hence, we formed four manifolds that reside in R^3 to construct this multi-manifold problem, and we wish to run our expectation maximization algorithms on this data set to retrieve similar looking clusters represented in R^2. To see why the construction of our expectation manifold algorithm is worthwhile, we have included the outcome of running the standard EM algorithm on our dataset.



Visually, this clustering does not seem right. A human observer would expect the different clusters to be the four different circular regions. In fact, we know that this is the proper clustering, because our generated manifold comes from the original cluster data in R^2 that we transformed into the R^3 space. At first, our implementation of the expectation maximization algorithm was not ideal in terms of clustering results, and the output was significantly contingent on the initial cluster centers. The related figures have been shown and discussed in section 3.1 Challenges. This prompted us to search for new ways of cluster initialization (as noted in the challenges subsection of section 3). With the implementation of our nearest neighbor merge algorithms we were able to consistently achieve favorable clusterings. Using the minimum nearest neighbor merge variant of our NN-merge algorithm we get:



As before, the red triangles indicate the initial cluster centers. Our min nearest neighbor merge algorithm managed to achieve our goal of initializing the cluster centers in a manner that puts each of the k initial centers into the distinct k clusters (as we have constructed the problem). Below is the output of our expectation maximization algorithm running given these above points:



This clustering is almost identical to the original set of data points in R^2 which we transformed to create our manifold. There is some excess purple points where green points should ideally be; however, it is significantly better than the standard EM algorithm or having a random set of initial cluster centers.

Algorithm Used	Purity Measure
Traditional Expectation Maximization	0.53
Spectral Clustering	0.36
Minimum NN Merge EM	0.996

This table shows some results on the manifold dataset described above. We can see how the traditional expectation maximization algorithm and spectral clustering (another clustering algorithm we used to compare) have a significantly lower purity measure compared to that of our expectation maximization algorithm when using the minimum nearest neighbors merge initialization algorithm.

5. Discussion

We note that the purity measure of our algorithm when applied with the min merge algorithm is exceptionally high (0.996) can indicate some sort of bias with respect to our construction of the problem. However, it does make sense that we are able to achieve near perfect purity (a purity of 1) because we know for a fact that an underlying cluster structure exists on our manifold since we made sure of it when we transformed the clustered data from R^2 to the manifold in R^3. Furthermore, we also acknowledge that the K-means algorithm works extremely well on our dataset as well, giving near identical results to our min and merge algorithms. This indicates that our algorithm may actually not indicate bias. It also may indicate that our algorithm is not more powerful than the k-means algorithm, so it is logical to question the significance of our algorithm as a means to solve our original clustering problem on a manifold. On the other hand, our algorithm has problems working with data matrices of low rank and high sparsity. Specifically, our algorithm has trouble working on the MNIST dataset for its high dimensionality and the reasons listed just prior. A potential direction is to solve this problem by the means of constructing a manifold in a similar manner, but through data in lower dimension that is sparse and then transforming this data into a much higher dimension manifold. From this point, we would run the same EM algorithm experiments we have done in this research report on the sparser data. Ideally, this will give some indication of how well our algorithm can perform on low rank, high sparsity data matrices in high dimensional data (and then our algorithm may be extended to MNIST as a consequence). Another problem observed in our algorithm is that our algorithm sometimes returns fewer clusters than asked for due to numeric properties of the matrix we cannot control, including stationary points that exist within the dataset we can accidentally converge upon. This problem is also witnessed in the traditional expectation maximization algorithm.

6. Conclusion

We acknowledge that further research and experimentation with our algorithm is necessary to have more results that generalize. Our algorithm does phenomenally with respect to the purity measure; however, such a high value can also indicate some bias or overfitting, as we tried many different initialization algorithms to achieve such favorable results. Nonetheless, we believe we have implemented an algorithm such that it combines interesting features of Gaussian Mixture Modeling, Expectation Maximization, and Manifold learning that justifies further research and experimentation, as noted in the discussion section (i.e. dealing with low rank, high sparsity, high dimensional data).

References

[1] Wu, Yiming, and Kap Luk Chan. "An extended isomap algorithm for learning multi-class manifold." Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on. Vol. 6. IEEE, 2004.

[2] Zhu, Chunhui, Fang Wen, and Jian Sun. "A rank-order distance based clustering algorithm for face tagging." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[3] Souvenir, Richard, and Robert Pless. "Manifold clustering." Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 1. IEEE, 2005.

[4] Stanford University, "Evaluation of Clustering"

[5] Archambeau, Cédric, John Aldo Lee, and Michel Verleysen. "On Convergence Problems of the EM Algorithm for Finite Gaussian Mixtures." ESANN. Vol. 3. 2003.