

Project 1.1: Foram Segmentation

Amit Rao
NC State University
Raleigh, NC
arao4@ncsu.edu

Prathamesh Prabhudesai
NC State University
Raleigh, NC
ppprabhu@ncsu.edu

Siddharth Roheda
NC State University
Raleigh, NC
sroheda@ncsu.edu

Abstract

Image segmentation plays an important role in information extraction. This project focuses on the use of machine learning techniques to segment regions of Foraminifera images into one of five classes, viz. 'Background', 'Chamber', 'Edge between Background and Shell', 'Edge between Chambers', and 'Aperture'. The report compares PCA and Forward Feature Selection Technique for Feature Reduction. It further compares the performance of Support Vector Machine with a Supervised Clustering technique for classification. The best performance is obtained with SVM-FFSI technique, with an F1-Score of 0.7039.

Keywords: Foraminifera, Feature Reduction, Principal Component Analysis, Forward Feature Selection, Support Vector Machines, Supervised Clustering

1. Introduction

Foraminifera (Forams for short) are single-celled protists with shells. The shells are commonly divided into chambers, which are added during growth. Depending on the species, the shell may be made of organic compounds, sand grains and other particles cemented together, or crystalline calcite [1].

1.1. Dataset

The provided training dataset is spread over 40 different Forams. Each Foram is described by 8 lighting conditions, which are further divided into cells of 40 pixels. Each cell in a given lighting condition is described by a set of 10 features, leading to a set of 80 features per cell for a given Foram. Figure 1 shows the first Foram in the dataset, in all 8 lighting conditions.

The 10 features describing each cell, in each lighting condition are listed below:

1. mean_scale40 Mean intensity value over a neighborhood of size 40









- | Original images | | | |
|---|--|--|--|
| Image 1 | Image 2 | Image 3 | Image 4 |
|  |  |  |  |
| Image 5 | Image 6 | Image 7 | Image 8 |
|  |  |  |  |
2. std_scale40 Standard deviation of intensity values over a neighborhood of size 40
 3. FX_scale40 Gradient in the x-direction using a smoothing kernel of scale proportion to 40
 4. FY_scale40 Gradient in the y-direction using a smoothing kernel of scale proportion to 40
 5. magnitude_scale40 Magnitude of gradient
 6. mean_scale80 Mean intensity value over a neighborhood of size 80
 7. std_scale80 Standard deviation of intensity values over a neighborhood of size 80
 8. FX_scale80 Gradient in the x-direction using a smoothing kernel of scale proportion to 80
 9. FY_scale80 Gradient in the y-direction using a smoothing kernel of scale proportion to 80
 10. magnitude_scale80 Magnitude of gradient

Figure 1. 8 lighting conditions of a sample Foram

Figure 2 shows an example of these features for the first Foram in the first lighting condition.

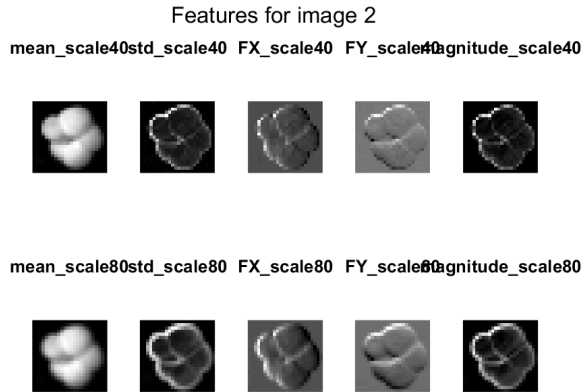


Figure 2. 10 features for one lightning condition of a sample Foram

In this project, we first select a subset of features from the larger set of 80 features, and then use this subset for creating a classification model. This model is then further used for classification of cells of Forams, and evaluated based on the F-score, Precision, and Recall [2].

The rest of the report is divided into three sections, namely, the Approach, Results, and Conclusion. The Approach section, which is the major part of this report, describes the problem and explains how to select a subset of features, and further discusses the implementation of classifiers for the classification of cells. The next section presents and discusses the results obtained from the simulations which were run using MATLAB. Finally, the Conclusion section talks about what was learned from the project and concludes it.

2. Approach

The available data has a total of 80 features for each sample. The first step is to reduce the number of features, by either feature reduction or selection of a subset from the set of original features. Both the methods are discussed in this section. Next, the test cells are classified using a classifier. In this project, we compare performance of the SVM classifier and a Supervised Clustering algorithm.

2.1. Feature Selection

Two feature selection methods are implemented and compared. The first method evaluated in this report is the Principal Component Analysis, and the second method is Forward Feature Selection. These methods are further discussed in the following subsections.

2.1.1 Principal Component Analysis

Principal component analysis (PCA) is a statistical procedure that uses the results from the Karhunen-Loeve transform to reduce the dimensions of a given data set by making use of variables called principal components. These principal components are nothing but the Eigen vectors of the covariance matrix of the data. In our case, we originally had 80 features, so we will get a covariance matrix of 80x80. After performing Eigen Value Decomposition, we get 80 Eigen vectors, as expected. The major Eigen vector defines the direction of the best fitting line, and is expected to give best separation in 1-D space. We select the top 12 Eigen vectors to project the data onto a 12 dimensional space. A 2-D to 1-D reduction is illustrated in figure 3.

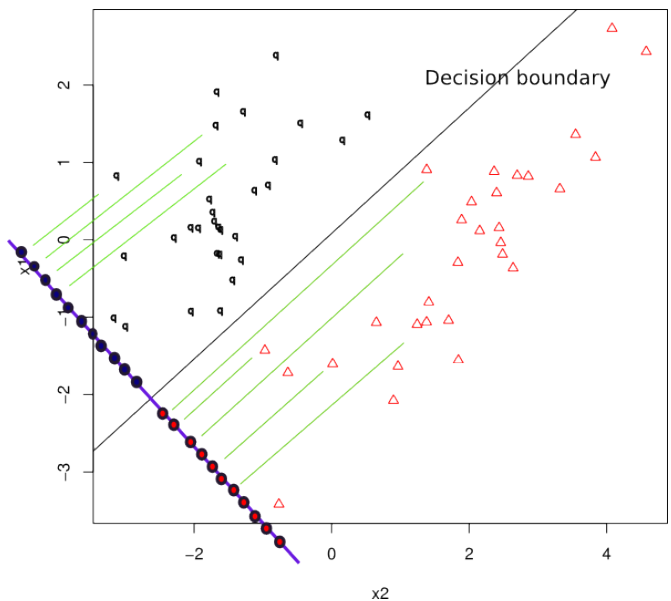


Figure 3. Projection using PCA

2.1.2 Forward Feature Selection

Before creating a training model, a subset of features must be selected from the 80 features that are available. This is done so as to enhance the generalization and avoid overfitting. In this project, a **forward feature selection** technique is also evaluated in order to select the most significant features.

In the first iteration, all the 80 features are evaluated individually, using the target classifier, and a k-cross validation ($k = 10$). The feature with the best performance is selected, call it x_1 , and added to an array of selected features. In the next iteration, all possible combinations of pairs of features, where the first feature is x_1 and the second feature is selected from the remaining 79 features are evaluated. The best combination is then added to the array, say x_1, x_2 . The

performance of the classifier with features x_1, x_2 is compared with the performance of the classifier with x_1 , and x_2 is only accepted if the performance with x_1, x_2 is superior. The process continues until an optimum combination is found.

Generalizing, in the i^{th} iteration, the best performing feature, x_i is selected based on the performance with the target classifier. x_i is added to the array of selected features only if the performance of the classifier using $x_1, x_2, \dots, x_{i-1}, x_i$ is better than x_1, x_2, \dots, x_{i-1} , and further iterations are performed as long as $i < N$, where N is the total number of features. The selection process is terminated if the performance is found to be inferior. For this method, a subset of 10 features was obtained. This feature extraction method is referred to as **FFS-1** in the rest of the report.

The above described method has a very high time complexity. In order to reduce the time complexity, an assumption of the features being independent is made. This means, only one iteration is evaluated, and the top 12 features are selected. This method is referred to as **FFS-2** in the rest of the report. The performance of both approaches is further compared in table 1.

2.2. Classification

Two methods are implemented in this project in order to classify cells into one of the five classes. The first method uses **Support Vector Machines** while second method uses **Supervised Clustering** for classification. These methods are further discussed in the subsections below.

2.2.1 Support Vector Machines

Figure 4 shows a case for separable data, where two possible hyper-planes are shown. Both of these planes correctly separate the data and are viable options for a classifier. But, the dotted plane is a better option, since it is more generalized. SVM selects the most generalized case from all the possible cases.

SVM is also known as a **maximum margin classifier**, since it maximizes the difference vector $x^+ - x^-$, as seen in figure 5, in order to select the optimal hyper-plane.

In case of non-linear data, a kernel functional must be used to transform the features into a linear space. Data in this project, as is the case with most real world data, is non-linear. We have used the radial basis function to transform the data into a higher dimensional space, where the data is linearly separable. The radial basis functional is defined as in equation 1.

$$K(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}} \quad (1)$$

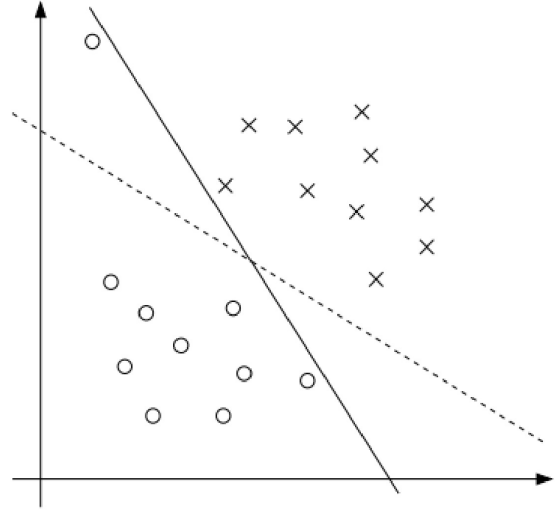


Figure 4. Possible for separable data

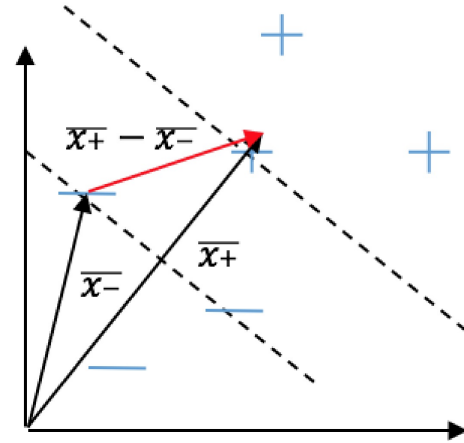


Figure 5. Possible for separable data

2.2.2 Supervised Clustering

The training set of clusters are interpreted as existing clusters, and the samples are to be assigned to one of these clusters. Distances from each data point in each cluster are calculated, and the data point with minimum distance is selected to represent each cluster. The test sample is assigned to the cluster with the closest cluster representative. The algorithm is represented in equation 2.

$$d(TS, CR_i) = \min(d(C_{ij}, TS)) \quad (2)$$

Where TS is the test sample, CR_i is the i^{th} cluster representative, and C_{ij} is the j^{th} sample of the i^{th} cluster.

The test sample is assigned to the cluster with $distance = \min(d(TS, CR_i))$

3. Results

The following methods have been evaluated in this report:

1. **PCA-SVM** - PCA for Feature Reduction, and SVM for Classification
2. **FFS1-SVM** - FFS1 (as described in section 2.1.2) for Feature Reduction, and SVM for Classification
3. **FFS2-SVM** - FFS2 (as described in section 2.1.2) for Feature Reduction, and SVM for Classification
4. **PCA-SC** - PCA for Feature Reduction, and Supervised Clustering for Classification
5. **FFS1-SC** - FFS1 (as described in section 2.1.2) for Feature Reduction, and Supervised Clustering for Classification
6. **FFS2-SC** - FFS2 (as described in section 2.1.2) for Feature Reduction, and Supervised Clustering for Classification

The dataset of 40 forams has been evaluated using a k-fold cross-validation technique, with $k = 4$.

3.1. Segmentation Results

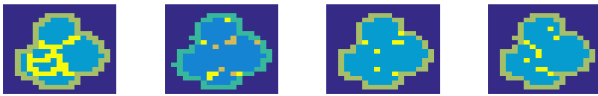


Figure 6. Ground Truth for Foram 38 on the left, Second from left: PCA-SVM segmentation, Third from left: FFS1-SVM segmentation, and FFS2-SVM segmentation on the right



Figure 7. Ground Truth for Foram 38 on the left, Second from left: PCA-SC segmentation, Third from left: FFS1-SC segmentation, and FFS2-SC segmentation on the right

Figure 6 shows the Ground Truth for the 38th Foram in the provided dataset, and compares it with the results of PCA-SVM, FFS1-SVM and FFS2-SVM methods.

Figure 7 shows the Ground Truth for the 38th Foram in the provided dataset, and compares it with the results of PCA-SC, FFS1-SC and FFS2-SC methods.

3.2. Precision, Recall, and F-Score

3.2.1 Definitions and Formulas

1. Precision: the number of correctly classified positive examples divided by the number of examples labeled by the system as positive [2].

$$Precision = \frac{tp}{tp + fp} \quad (3)$$

2. Recall: the number of correctly classified positive examples divided by the number of positive examples in the data [2].

$$Recall = \frac{tp}{tp + fn} \quad (4)$$

3. Fscore: a combination of the above [2].

$$F_{\beta} - Score = \frac{(\beta^2 + 1)tp}{(\beta^2 + 1)tp + \beta^2 fn + fp} \quad (5)$$

$$F_{\beta} - Score = \frac{(\beta^2 + 1)Precision_M * Recall_M}{\beta^2 Precision_M + Recall_M} \quad (6)$$

Method	Avg Prec	Avg Rec	Avg F_1 - Score	Max F_1 - Score
PCA-SVM	0.6347	0.6188	0.6270	0.7288
FFS1-SVM	0.7318	0.6616	0.7039	0.8419
FFS2-SVM	0.7255	0.6544	0.6889	0.8005
PCA-SC	0.6176	0.5986	0.6077	0.6511
FFS1-SC	0.6667	0.6174	0.6408	0.6732
FFS2-SC	0.6350	0.6004	0.6170	0.6384

Table 1. Comparison of Precision, Recall, and F1-score for all implemented methods

4. Conclusion

In this project, combinations of three different feature reduction algorithms with two classification techniques have been evaluated. Of the six possible combinations, the best performance was obtained with the combination of FFS1 (Forward Feature Selection without Independence assumption) approach for feature reduction and SVM for classification. FFS1, in spite of being computationally costly, is a one time cost, and hence, affordable. The combination of FFS1 and SVM gives an average F1-Score of 0.7039.

References

- [1] <http://www.ucmp.berkeley.edu/foram/foramintro.html>, Accessed: 10/06/2016
- [2] M. Sokolova and G. Laplame, "A Systematic Analysis of Performance Measures for Classification Tasks", Information Processing and Management 45, 2009.
- [3] S. Theodoridis and K. Koutroubas, Pattern Recognition.
- [4] <http://cs229.stanford.edu/notes/cs229-notes5.pdf>

Project 1.2: Proposal for Markov Network based Image Segmentation

Amit Rao
NC State University
arao4@ncsu.edu

Prathamesh Prabhduesai
NC State University
ppprabhu@ncsu.edu

Siddharth Roheda
NC State University
sroheda@ncsu.edu

Abstract

In this report, we propose an image segmentation algorithm based on Markov Random Fields(MRF). Images are modeled using MRF and probabilities of true labels given the features are calculated using Energy Minimization which includes the weighted node and edge potentials. The report further discusses additional features that can be used for the segmentation procedure.

Keywords: Markov Random Field, Hammersley-Clifford, Expectation Maximization

1. Introduction

Foraminifera (Forams for short) are single-celled protists with shells. The shells are commonly divided into chambers, which are added during growth. Depending on the species, the shell may be made of organic compounds, sand grains and other particles cemented together, or crystalline calcite [1].

In the previous project, traditional machine learning algorithms were implemented to label regions in the Forams. Feature reduction techniques such as Principal Component Analysis, Forward Sequential Selection and Correlation based feature selection were implemented to avoid overfitting of the data and this reduced feature list was then used for training the classifier viz. Support Vector Machine and Minimum Distance Supervised Clustering. An average F1 score of 0.7039 was achieved using Support Vector Machine and 0.6567 for Supervised Clustering . To further improve the accuracy of the Foram segmentation, we propose an algorithm based on *Undirected Graphical Models*.

2. Approach

This section discusses the algorithm for the specific region segmentation using Markov Random Field.

2.1. Pixel Labeling Task

Each pixel (*superpixel* in our case) has a set of features which defines the class for that pixel. Let s represent a sin-

gle superpixel in our image having a feature vector f_s associated with it. Then for the whole image we have,

$$\mathbf{f} = \{\vec{f}_s : s \in S\} \quad (1)$$

where, S is the set of all superpixels defining the image and \mathbf{f} is the vector of features defining this complete image. \vec{f}_s includes reduced set of features selected for the local classifiers in project 1.1 along with additional features, that will be defined later in this report.

As we know that each superpixel is given some label ω_s from the predefined set of labels, $\Lambda = \{0, 1, 2, 3, 4\}$. So for the whole image we have,

$$\Omega = \{\omega_s, s \in S\} \quad (2)$$

2.2. Maximum A-Posteriori (MAP) Approach for Label Assignment

The idea here is to define a probability measure on the set of all possible labeling and select the most likely one. $P(\Omega|\mathbf{f})$ measures the probability of a labeling, given the observed feature vector \mathbf{f} . The goal here is to find an optimal labeling $\hat{\Omega}$ which maximizes $P(\Omega|\mathbf{f})$. This is called as the MAP estimate.

$$\hat{\Omega}_{MAP} = \underset{\Omega}{\operatorname{argmax}} P(\Omega|\mathbf{f}) \quad (3)$$

From Bayes Theorem, we have

$$p(\Omega|\mathbf{f}) = \frac{P(\mathbf{f}|\Omega)P(\Omega)}{P(\mathbf{f})} \quad (4)$$

Since $P(\mathbf{f})$ is going to be constant, we can say that

$$P(\Omega|\mathbf{f}) \propto P(\mathbf{f}|\Omega)P(\Omega) \quad (5)$$

So now we need to define $P(\Omega)$ and $P(\mathbf{f}|\Omega)$ to get the Probability distribution of the labels in our model.

2.3. Markov Random Field for Modeling

In real images, regions are often homogeneous; neighboring pixels usually have similar properties (intensity, color, texture etc.). MRF is a probabilistic model which captures such contextual constraints [4].

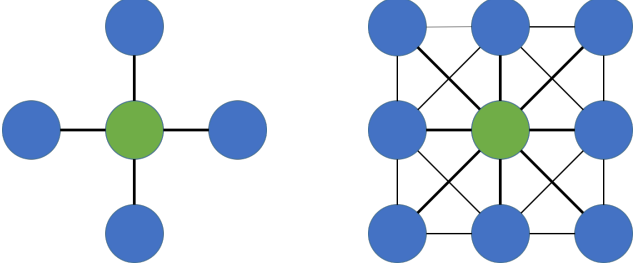


Figure 1. First Order Neighborhood on left, and Second Order Neighborhood on right

Given an undirected graph $G(V,E)$ where V are the set of vertices and E the edges connecting the vertices, a set of random variables $X = (X_v)$ where $v \in V$ indexed by V , form a Markov random field with respect to G if they satisfy the following properties [5]:

- **Pairwise Markov Property:** Any two non-adjacent variables are conditionally independent given all other variables.
- **Local Markov Property:** A variable is conditionally independent of all other variables given its neighbors.
- **Global Markov Property:** Any two subsets of variables are conditionally independent given a separating subset.

To model the image using MRF, we need some basic building blocks; such as Observation Field and Labeling Field (hidden), Pixels and their Neighbors, Cliques and Clique Potentials, Energy Function, Gibbs Distribution.

2.4. Neighborhood

For each pixel, we can define some surrounding pixels as its neighbors. Neighborhood can be defined in many ways such as 1st order neighborhood, 2nd order neighborhood, as shown in figure 1.

2.5. Hammersley-Clifford Theorem

This theorem states that a random field is considered an MRF if and only if $P(\Omega)$ follows a Gibbs Distribution. Therefore,

$$P(\Omega) = \frac{1}{Z} \exp(-U(\Omega)) = \frac{1}{Z} \exp\left(-\sum_{c \in C} V_c(\omega)\right) \quad (6)$$

where $Z = \sum \exp(-U(\Omega))$ is a normalization constant; C represents a Clique and V_c is the clique potential. So this theorem provides us an easy way of defining MRF models via Clique Potentials.

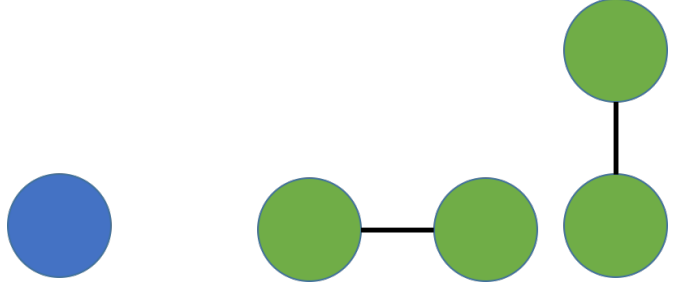


Figure 2. Clique corresponding to node potential on left, and clique corresponding to edge potential on right

2.6. Clique and Clique Potential

A subset $C \subset S$ is called a clique if every pair of pixels in the subset are neighbors. A clique containing n pixels is called n^{th} order clique and usually denoted as C_n . The set of cliques in an image is denoted by $C = C_1 \cup C_2 \cup \dots \cup C_k$ [2].

For each clique c in the image, we can assign a value $V_c(\omega)$ which is called as Clique Potential of c , where ω is the configuration of the labeling field. From the equation (6), we know that sum of potentials of all cliques gives us the energy $U(\Omega)$ of the configuration Ω .

$$U(\Omega) = \sum_{c \in C} V_c(\omega) \quad (7)$$

$$U(\Omega) = \sum_{i \in C_1} V_{C_1}(\omega_i) + \sum_{(i,j) \in C_2} V_{C_2}(\omega_i, \omega_j) \quad (8)$$

where V_{C_1} is the node potential and V_{C_2} is the edge potential, and the cliques corresponding to them can be seen in figure 2. These potentials are defined in terms of features by defining V_{c_1} and V_{c_2} as follows:

$$V_{c_1} = w_{n_i} f_{n_i} \quad (9)$$

$$V_{c_2} = w_{e_{ij}} \|f_{e_i} - f_{e_j}\|_p \quad (10)$$

Here, the p^{th} norm is defined as, $\|\mathbf{x}\|_p = (\sum_i x_i^p)^{1/p}$, $x_i \in \mathbf{x}$. f_n and f_e are the set of features used for determining the node potentials and the edge potentials respectively, and w_n and w_e are the corresponding weight vectors.

3. Parameter Estimation and Energy Minimization

The target is to find the most probable labeling i.e. we want to find the Ω that maximizes $P(\Omega|f)$, as shown in equation (11), which can also be achieved by minimizing the energy function in equation (13).

$$P(\Omega|f) = \frac{1}{Z} e^{(-\sum_{i \in c_1} w_{n_i} f_{n_i})} e^{(-\sum_{(i,j) \in c_2} w_{e_{ij}} \|f_{e_i} - f_{e_j}\|_p)} \quad (11)$$

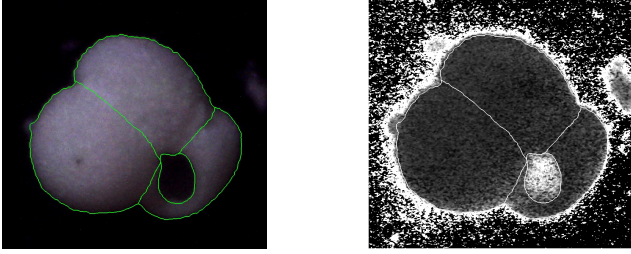


Figure 3. Original Image on Left, and Saturation Image on Right

$$\hat{\Omega}_{MAP} = \underset{\Omega}{\operatorname{argmax}} P(\Omega|f) \quad (12)$$

$$U(\Omega|f) = \sum_{i \in C_1} (w_{n_i} f_{n_i}) + \sum_{(i,j) \in C_2} (w_{e_i} \|f_{e_i} - f_{e_j}\|_p) \quad (13)$$

$$\hat{\Omega}_{MAP} = \underset{\Omega}{\operatorname{argmin}} U(\Omega|f) \quad (14)$$

Before this approach can be used to segment an input image, the parameters for this formulation must be estimated, i.e. the weight vectors w_n and w_e . This can be done by finding the weight vectors that minimize the energy function for the labeling over the training data. This is a non-convex optimization problem. This can be solved using Simulated Annealing for minimizing the energy function in (13). Another approach to estimating the weight vectors is to use Expectation-Maximization on equation (11) as discussed in [3].

4. Classification Features

The features used in the vector f_n are the same as the ones that were obtained after Forward Feature Selection in project 1.1. For the feature vector f_e , we suggest the use of two features in addition to the features from project 1.1.

4.1. Local Classifier Output

The output from the local classifier implemented in project 1.1 can be used as a feature for the edge potential, i.e. in the feature vector f_e . Since we used multiple classifiers in the previous project, some combinational approach such as, maximum voting, or Dempster-Shafer fusion, may be used to obtain an optimized Local Classifier output. Using this as a feature ensures smoothness of the final classification.

4.2. Saturation

Another additional feature proposed here for the feature vector f_e is saturation. It is seen in figure 3 that saturation provides a good separation between aperture, chamber, and background. Hence, may be able to improve the segmentation.

References

- [1] <http://www.ucmp.berkeley.edu/foram/foramintro.html>, Accessed: 10/06/2016
- [2] https://inf.u-szeged.hu/ssip/2008/presentations2/Kato_ssip2008.pdf, Accessed: 11/02/2016
- [3] Quan Wang, HMRF-EM-image: Implementation of the Hidden Markov Random Field Model and its Expectation-Maximization Algorithm, December 2012
- [4] Simon A. Barker, Image Segmentation using Markov Random Field Models, Dissertation July 1998, University of Cambridge
- [5] https://en.wikipedia.org/wiki/Markov_random_field, Accessed: 11/02/2016
- [6] Daphne Koller and Nir Friedman, Probabilistic Graphical Models: Principles and Techniques

Image Segmentation using Conditional Random Fields

Amit Rao
NC State University
Raleigh, NC
arao4@ncsu.edu

Prathamesh Prabhudesai
NC State University
Raleigh, NC
ppprabhu@ncsu.edu

Siddharth Roheda
NC State University
Raleigh, NC
sroheda@ncsu.edu

Abstract

Image segmentation plays an important role in information extraction. This project focuses on the use of Undirected Graphical Models (UGM) to segment regions of Foraminifera images into different regions. Image modeling is done as a Conditional Random Field as well as Hidden Markov Model. The highest F1 score obtained was 0.7365 with an overlap ratio of 0.8997.

Keywords: Foraminifera, Conditional Random Field, Undirected Graphical Model, Hammersly-Clifford, Hidden Markov Model, Overlap Ratio

1. Introduction

Foraminifera (Forams for short) are single-celled protists with shells. The shells are commonly divided into chambers, which are added during growth. Depending on the species, the shell may be made of organic compounds, sand grains and other particles cemented together, or crystalline calcite [1]. In the previous part of this project (Project 1.1), we used local classifiers such as Support Vector Machine (SVM) and Fine K-Nearest Neighbor (kNN with $k = 1$). In this part of the project we intend to improve the performance of classification and segmentation by modeling the images using Conditional Random Field (CRF).

1.1. Dataset

The provided training dataset is spread over 40 different Forams. Each Foram is described by 8 lighting conditions, which are further divided into cells of 40 pixels. Each cell in a given lighting condition is described by a set of 10 features, leading to a set of 80 features per cell for a given Foram. Figure 1 shows the first Foram in the dataset, in all 8 lighting conditions.

The 10 features describing each cell, in each lighting condition are listed below:

1. mean_scale40 Mean intensity value over a neighborhood of size 40

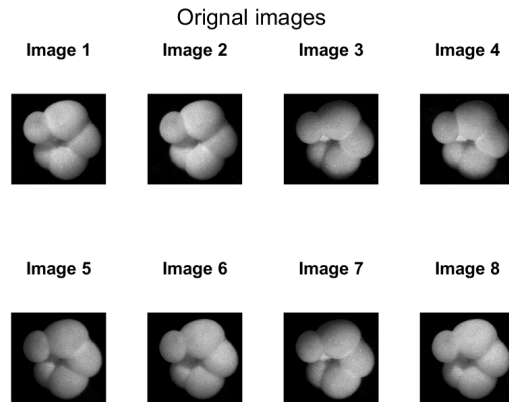


Figure 1. 8 lighting conditions of a sample Foram

2. std_scale40 Standard deviation of intensity values over a neighborhood of size 40
3. FX_scale40 Gradient in the x-direction using a smoothing kernel of scale proportion to 40
4. FY_scale40 Gradient in the y-direction using a smoothing kernel of scale proportion to 40
5. magnitude_scale40 Magnitude of gradient
6. mean_scale80 Mean intensity value over a neighborhood of size 80
7. std_scale80 Standard deviation of intensity values over a neighborhood of size 80
8. FX_scale80 Gradient in the x-direction using a smoothing kernel of scale proportion to 80
9. FY_scale80 Gradient in the y-direction using a smoothing kernel of scale proportion to 80
10. magnitude_scale80 Magnitude of gradient

Figure 2 shows an example of these features for the first Foram in the first lighting condition.

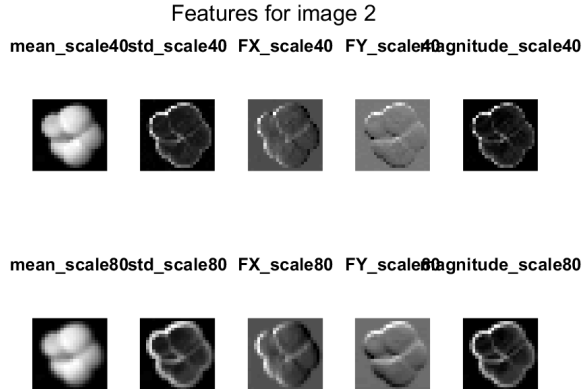


Figure 2. 10 features for one lighting condition of a sample Foram

There are two types of Ground Truth tables provided for this data. The old data set has 5 classes viz. background (class 0), edges between background and chamber (class 1), chambers (class 2), edge between chambers and aperture (class 4) and aperture (class 5). The new data set has the classes based on the number of regions in the image viz. background (class 0), aperture (class 1) and chambers (class 2 and above). We are considering all the chambers in the same class since the feature values are not going to be drastically different (class 2).

In this project, we used the subset of features obtained from Forward Feature Selection (FFS) technique implemented in the previous part of the project and further used these features to implement a Conditional Random Field and learn the parameters. This model is then further used for classification of cells of Forams, and evaluated based on the Precision, Recall, F-Score [2] and Overlap Ratio [1].

The rest of the report is divided into three sections, namely, the Approach, Results, and Conclusion. The Approach section, which is the major part of this report, describes the problem and explains how to learn the parameters for Conditional Random Field and further segments the Test images. The next section presents and discusses the results obtained from the simulations which were run using MATLAB. Finally, the Conclusion section talks about what was learned from the project and concludes it.

2. Approach

In order to improve the segmentation performance, we use Conditional Random Fields (CRF), instead of local classifiers. In order to implement CRF, the UGM toolbox is used, as suggested in the proposal by group 1 [3], and can

be found for download at [4].

2.1. Conditional Random Fields

Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. The primary advantage of CRFs over hidden Markov models is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. Additionally, CRFs avoid the label bias problem, a weakness exhibited by maximum entropy Markov models (MEMMs) and other conditional Markov models based on directed graphical models. CRFs outperform both MEMMs and HMMs on a number of real-world tasks in many fields, including bioinformatics, image segmentation, computational linguistics and speech recognition.

2.1.1 Hammersly-Clifford Theorem

We use the Hammersly-Clifford theorem to get the label (Ω) probability, given the feature vector (F). We consider node potentials, and pairwise clique potentials (edge potentials) to be non-negative, and all other clique potentials to be zero. The Hammersly-Clifford theorem is then given as,

$$P(\Omega|F) = \frac{1}{Z} \exp\left\{-\left(\sum_{i \in C_1} V_{C_1}(\omega_i) + \sum_{(i,j) \in C_2} V_{C_2}(\omega_i, \omega_j)\right)\right\} \quad (1)$$

Here, $V_{C_1}(\omega_i)$ are the node potentials for the i^{th} node, $V_{C_2}(\omega_i, \omega_j)$ are the edge potentials for the edge between the i^{th} and j^{th} node, and Z is the normalizing constant.

The Node and Edge Potentials are further defined in terms of parameters, w_{n_i} and $w_{e_{ij}}$, and the features, $F = \{f_i\}$, as,

$$V_{C_1} = w_{n_i} f_{n_i} \quad (2)$$

$$V_{C_2} = w_{e_{ij}} \|f_{e_i} - f_{e_j}\|_p \quad (3)$$

Where w_n is the vector of parameters for node potentials, and w_e is the vector of parameters for edge potentials. Further, the features, $F = \{f_i\}$ is a vector of 12 features, as selected in project 1.1, using Forward Feature Selection. Also, we add a 13th feature, which is the output of the local classifier (SVM) from project 1.1.

Before being able to implement the above discussed approach, the parameters, w_n and w_e must be learned. In this project, Mean Field Annealing is used to find the parameters that minimize the function $P(\Omega|F)$, equation ??, over

the training set. Mean Field Annealing is further discussed in the following section.

$$[w_n, w_e] = \min_{\{w_n, w_e\}} \left\{ \frac{1}{Z} e^{(-\sum_{i \in c_1} w_{n_i} f_{n_i})} e^{(-\sum_{(i,j) \in c_2} w_{e_{ij}} \|f_{e_i} - f_{e_j}\|_p)} \right\} \quad (4)$$

2.1.2 Mean Field Annealing

Mean Field annealing (MFA) is a technique for finding a good minimum of complex functions which typically have many minima. The mean field approximation of statistical mechanics allows a continuous representation of the energy states of a collection of particles. In the same sense, MFA approximates the stochastic algorithm called simulated annealing. Because computations using the mean transitions attain equilibrium faster than those using the corresponding stochastic transitions, mean field annealing relaxes to a solution at each temperature much faster than does stochastic simulated annealing. This leads to a significant decrease in computational effort.

The minimization function (minFunc) in the UGM toolbox provides an option (@UGM_Mean_Field) to use mean field annealing for estimation of parameters

2.1.3 Labeling

The UGM Toolbox provides the function UGM.Decode_* for finding the labels with maximum probability for the testing images. The function takes in node potentials, edge potentials, and the graph structure as input. We used different algorithms for decoding such as Iterated Conditional modes (ICM) and Linear Programming (LinProg).

Iterated Conditional Modes

To maximize the joint probability of a CRF this deterministic algorithm is used which maximizes local conditional probabilities sequentially. The ICM algorithm uses the *Greedy* strategy in the iterative local maximization. It makes two basic assumptions, one based on the contents of images in general and another based on the noise process having nothing to do with the image properties. The first assumption says that neighboring pixels tends to have the same values while the second assumption says each pixel is corrupted (flipped) independently, and with some probability.

In the basic algorithm, we initialize the nodes to some starting state values (the states that maximizes the node potentials) and we then start cycling through the nodes in order. When we get to node i , we consider all states that node i could take and replace its current state with the state that maximizes the joint potential. We keep cycling through the

nodes in order until we complete a full cycle without changing any nodes. At this point we reached a local optima of the joint potential that cannot be improved by changing the state of any single node. In this project we have used ICM with restart which is an effective enhancement of ICM algorithm. It suggests that whenever ICM reaches a local optima restart the optimization with a different initial configuration. By doing this we may be able to explore different local optima. After enough restarts and a guarantee that our restart mechanism has generated all the possible configurations, this procedure will eventually find the global optima.

Linear Programming

A linear programming problem may be defined as the problem of maximizing or minimizing a linear function subject to linear constraints. In case of CRFs the problem of computing the optimal decoding can be formulated as a binary integer linear program and solving this integer program will yield the optimal decoding. Unfortunately solving integer program is NP-Hard and the time required to complete is unpredictable. By relaxing the integer constraints we can obtain a polynomial-time approximate decoding. In particular, rather than enforcing the solution in the set $\{0,1\}$, we enforce that it is in the interval $[0,1]$. This makes all the constraints linear so combined with the linear objective function this becomes a linear program.

The linear programming relaxation may not always yield integer solutions however it is known that whenever some part of the solution is an integer that part of the solution must be the part of optimal decoding. Further, ignoring the issues of ties the entire solution will be integers with attractive potentials.

2.2. Alternative Approach: Hidden Markov Models

Another way to look at this problem of image segmentation, is to look at it as a sequence of super-pixels. Looking at the super-pixels as a sequence, a Transition and Emission matrix may be trained, like in project 2.2. The Transition and Emission may be further used to predict the states, given a set of observations, using the Matlab function, hmdecode(). The graph structure in this case would be more like a directed chain (sequence), rather than what is discussed in section 2.1.

We have 12 features (selected from the initial 80 features using Forward Feature Selection) describing each state (0, 1, 2, 3, 4). These features are the observations in the modeling of the Hidden Markov Model. Instead of handling 12 observations for each state, it would be easier to model a single representative observation, which somehow describes these 12 features. A good representative would be the classification result from the Local Classifier used for Project 1.1. We achieved the best performance with SVM

in project 1.1, and hence, we use SVM for obtaining the representative observation for the 12 features.

2.3. Evaluation Metric

We use the evaluation metric as discussed in [2]. In particular, we use $Precision_M$, $Recall_M$, and $Fscore_M$ for multi-class classification which are defined as below:

1. $Precision_M$: Precision is the agreement of the data labels with the positive labels generated by the classifier for each class. Thus, the $Precision_M$ for l -class classification is defined as:

$$Precision_M = \frac{\sum_{i=1}^l \frac{tp_i}{tp_i + fp_i}}{l} \quad (5)$$

2. $Recall_M$: Recall is the measure of quality of the classifier to correctly identify the labels. Thus, the $Recall_M$ for l -class classification is defined as:

$$Recall_M = \frac{\sum_{i=1}^l \frac{tp_i}{tp_i + fn_i}}{l} \quad (6)$$

3. $Fscore_M$: Fscore is the combination of precision and recall to measure. It is given as:

$$Fscore_M = \frac{(\beta^2 + 1)Precision_M * Recall_M}{\beta^2 Precision_M + Recall_M} \quad (7)$$

For our evaluation, we set $\beta = 1$, which is referred to as $F1score_M$.

3. Results

The following methods have been evaluated in this report:

1. **OD-LP** - Old data-set is evaluated, with Mean Field Annealing for CRF inference and parameter calculation, while for decoding Linear Programming is used. Results for each class can be seen in table 1, and average precision, recall, F1 score, and Overlap Ratio can be seen in table 6.

Class	Precision	Recall	F1 Score
0	0.9820	0.9596	0.9707
1	0.8681	0.7868	0.8255
2	0.8714	0.8887	0.8800
3	0.0827	0.1286	0.1007
4	0.2200	0.7700	0.3422

Table 1. Precision, Recall and F1 score for old data-set with Linear Programming as the decoding technique

2. **OD-ICM** - Old data-set is evaluated, with Mean Field Annealing for CRF inference and parameter calculation, while for decoding Iterated Conditional Modes is used.

Results for each class can be seen in table 2, and average precision, recall, F1 score, and Overlap Ratio can be seen in table 6.

Class	Precision	Recall	F1 Score
0	0.9685	0.9600	0.9642
1	0.7308	0.7889	0.7587
2	0.8757	0.8814	0.8785
3	0.2789	0.1802	0.2189
4	0.2286	0.7407	0.3493

Table 2. Precision, Recall and F1 score for old data-set with Iterated Conditional Modes as the decoding technique

3. **OD-HMM** - Old data-set is evaluated with Hidden Markov Models.

Results for each class can be seen in table 3, and average precision, recall, F1 score, and Overlap Ratio can be seen in table 6.

Class	Precision	Recall	F1 Score
0	0.9903	0.9924	0.9914
1	0.8359	0.9889	0.9060
2	0.8962	0.9500	0.9223
3	0.6471	0.2705	0.3815
4	0.8667	0.3333	0.4815

Table 3. Precision, Recall and F1 score for old data-set with Hidden Markov Model

4. **ND-LP** - New data-set is evaluated, with Mean Field Annealing for CRF inference and parameter calculation, while for decoding Linear Programming is used. Here, we evaluate results for 3 classes, 0 - Background, 1 - Aperture, and 2 - Chambers.

Results for each class can be seen in table 4, and average precision, recall, F1 score, and Overlap Ratio can be seen in table 6.

Class	Precision	Recall	F1 Score
0	0.9511	0.9511	0.9511
1	0.2215	0.2681	0.2426
2	0.9560	0.8955	0.9248

Table 4. Precision, Recall and F1 score for new data-set with Linear Programming as the decoding technique

5. **ND-ICM** - New data-set is evaluated, with Mean Field Annealing for CRF inference and parameter calculation, while for decoding Iterated Conditional Modes is

used. Here, we evaluate results for 3 classes, 0 - Background, 1 - Aperture, and 2 - Chambers.

Results for each class can be seen in table 5, and average precision, recall, F1 score, and Overlap Ratio can be seen in table 6.

Class	Precision	Recall	F1 Score
0	0.9529	0.9506	0.9517
1	0.2773	0.2261	0.2499
2	0.9550	0.8958	0.9245

Table 5. Precision, Recall and F1 score for new data-set with Iterated Conditional Modes as the decoding technique

Methods	$F1Score_M$	OR
OD-LP	0.6518	0.9127
OD-ICM	0.6601	0.8997
OD-HMM	0.7365	0.8997
ND-LP	0.7072	0.8809
ND-ICM	0.7091	0.8765

Table 6. Macro-F1 score and Overlap Ratio (OR) for all the algorithm on both old and new dataset

From the above results, it looks like the Hidden Markov Model approach outperforms all the other discussed approaches in terms of F1 score as a metric. But, OD-LP is a better performer in terms of Overlap Ratio of chambers. A likely reason for this is that, the OD-LP approach does well with detecting chambers (class 2), but struggles to detect edges between chambers (class 3), and hence bringing down the F1 score, but not the Overlap Ratio, which is evaluated for chambers (class 2).



Figure 3. Ground Truth for Foram 26 on the left, segmentation for OD-LP in the center, and segmentation using OD-ICM on the right

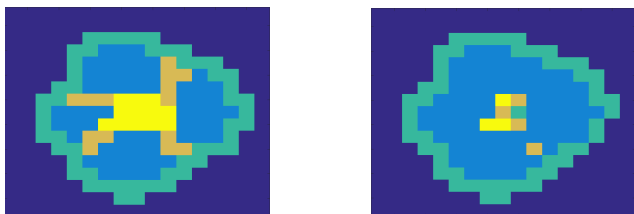


Figure 4. Ground Truth for Foram 34 on the left, and segmentation for the same foram using Hidden Markov Model

Classification of the new dataset images is considered to be a three class classification problem, class 0 for back-

ground, class 1 for aperture and class 2 for chambers. (class 2 and onward samples are considered to be of the same class since they all belong to the chamber class.) Various approaches for segmentation using Morphology, Filtering, Edge Preserving Smoothing, Connected Components are tried but no algorithm could separate out different regions of chamber.



Figure 5. Ground Truth for Foram 20 on the left, segmentation for ND-LP in the center, and segmentation using OD-ICM on the right

4. Conclusion

In this project, images have been modeled as Conditional Random Fields for implementing image segmentation. Mean Field Annealing was used to minimize the energy function which consists of the node and edge potentials. The weights then obtained are used for decoding using Iterated Conditional Modes and Linear Programming. This approach is further compared with Hidden Markov Model approach, and it is found that the Hidden Markov Model outperforms the CRF approach (OD-LP, and OD-ICM), in terms of F1 score. At the same time, it also provides a advantage in having a much lower computational complexity for training compared to CRF, which involves minimization of a cost function using Mean Field Annealing. Although, OD-LP gives a marginally better Overlap Ratio for class 2 (chambers), compared to HMM. A likely reason for this is that, the OD-LP approach does well with detecting chambers (class 2), but struggles to detect edges between chambers (class 3), and hence bringing down the F1 score, but not the Overlap Ratio, which is evaluated for chambers (class 2).

So, in terms of F1 score, the best performer is HMM, with an F1 score of 0.7365, while in terms of Overlap Ratio, the best performer is OD-LP, with 0.9127.

References

- [1] <http://www.ucmp.berkeley.edu/foram/foramintro.html>, Accessed: 10/06/2016
- [2] M. Sokolova and G. Laplame, "A Systematic Analysis of Performance Measures for Classification Tasks", Information Processing and Management 45, 2009.
- [3] Group 1, "Graphical Approach for Region Identification in Foraminifera Images"
- [4] Mark Schmidt. UGM: A matlab toolbox for probabilistic undirected graphical models.

<http://www.cs.ubc.ca/~schmidtm/Software/UGM.html>
[Accessed: 11/28/2016]

[5] S. Theodoridis and K. Koutroumbas, Pattern Recognition.