CHAPTER 3

MATHEMATICAL BACKGROUND OF LBP AND CLUSTERING APPROACHES

3.1 Introduction

The mathematical backgrounds of various techniques used for the proposed automated glaucoma analysis system are discussed in this chapter. The proposed system classifies the retinal fundus image into normal and glaucomatous based upon computed CDR value. The following sections describe the mathematical background of LBP, FCM, k-means clustering, and EM segmentation approaches. In this study, the proposed methodology considers several methodologies that have been done by previous researchers that relate to the thesis. To achieve better performance, the new technique has been proposed based on the methodologies given in this chapter.

3.2 Local Binary Pattern

LBP is simple, very discriminative, and computationally efficient local texture descriptors, which leads to significant progress in applying texture methods to various computer vision problems. It labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. The LBP method can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real world applications is its invariance against monotonic gray level changes caused by illumination variations. Another equally important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.
Ojala introduced original LBP operator, based on the assumption that texture has locally two complementary aspects, a pattern, and its strength. The operator works in a $3 \times 3$ neighbourhood, using the center value as a threshold. An LBP code is produced by multiplying the threshold values with weights given by the corresponding pixels, and summing up the result. As the neighbourhood consists of 8 pixels, a total of $2^8 = 256$ different labels can be obtained depending on the relative gray values of the center and the pixels in the neighbourhood. The contrast measure is obtained by subtracting the average of the gray levels below the center pixel from that of the gray levels above (or equal to) the center pixel. If all eight threshold neighbours of the center pixel have the same value (0 or 1), the value of contrast is set to zero. The distributions of LBP codes or 2-dimensional distributions of LBP and local contrast are used as features in classification or segmentation.

LBP operator utilizes a binary representation of texture units localized in image neighborhoods. It also represents the shape of the image (Ojala and Harwood 1996). This operator works with eight neighbors of a pixel, using the value of the center pixel as a threshold. If a neighbor pixel has a higher or equal gray value than the center pixel then ‘1’ is assigned to that pixel, else it gets ‘0’. Then assigned ones among eight neighbors of a pixel are multiplied by powers of two in clockwise or counter clock wise direction and then summed to obtain a pattern for the center. LBP features are computed by using Equation (3.1).

$$LBP(X_c, Y_c) = \sum_{n=0}^{7} 2^n S(i_n - i_c)$$

(3.1)

Where $i_c$ indicates the value of a central pixel is, $i_n$ represents the value of $N$ neighborhood pixels. Figure 3.1 shows the working process of LBP operator on eight neighborhood pixels.
3.3 Unsupervised Clustering

The development of image segmentation algorithms has drawn extensive and consistent attention in medical image analysis. Most evaluation methods are either subjective or tied to specific applications. The majority of supervised requires ground truth reference images for objective evaluation. The key advantage of unsupervised segmentation evaluation is that it does not require segmentations to be compared against a manually segmented reference image. This advantage is indispensable to general purpose segmentation applications, such as those embedded in real-time systems, where a large variety of images with unknown content and no ground truth need to be segmented. The ability to evaluate segmentations independently of a manually segmented reference image.
not only enables evaluation of any segmented image but also enables the unique potential for self-tuning. It would be preferable to have a self-tunable segmentation method that could dynamically adjust the segmentation algorithm's parameters to automatically determine the parameter options that generate better results. The unsupervised objective evaluation methods do not require a reference image for generating a segmentation evaluation metric.

Unsupervised evaluation is quantitative and objective. It has distinct advantages, perhaps the most critical of which is that it requires no reference image. A manually created reference image is intrinsically subjective and creating such a reference image is tedious and time-consuming, and for many applications, it is hard or maybe even impossible. The ability to work without reference images allows the unsupervised evaluation to operate over a wide range of conditions and with many different types of images. This property also makes unsupervised evaluation uniquely suitable for automatic control of online segmentation in real-time systems, where a wide variety of images, whose contents are not known beforehand need to be processed.

The information found in corresponding data is grouped into various data objects in clustering analysis module. The objective of the clustering analysis is a separation of objects into groups, with the similar objects together and unrelated with another group of objects. It is being applied by various expert research communities such optimization, computational geometry, machine learning and statistic. Figures 3.2 to 3.6 show the visual representation of the data clustering. Three unsupervised clustering approaches; \( k \)-means, FCM, and EM are exploited for OC segmentation using retinal fundus images. The mathematical backgrounds of these approaches are briefly discussed in this section.
Figure 3.2 Example of clustering visualization (#cluster = 1)

Figure 3.3 Example of clustering visualization (#cluster = 2)
Figure 3.4 Example of clustering visualization (#cluster = 3)

Figure 3.5 Example of clustering visualization (#cluster = 4)
3.3.1 k-Means Clustering

The k-means clustering algorithm is an iterative technique that is used to partition an image into k clusters. In statistics and machine learning, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The basic algorithm is:

- Pick k cluster centers, either randomly or based on some heuristic.
- Assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster center.
- Re-compute the cluster centers by averaging all of the pixels in the cluster.
- Repeat last two steps until convergence are attained (e.g. no pixels change clusters)
Given a set of observations \((x_1, x_2, \ldots, x_n)\), where each observation is a \(d\)-dimensional real vector, \(k\)-means clustering aims to partition then observations into \(k\) sets \((k < n) S = \{S_1, S_2, \ldots, S_k\}\) so as to minimize the within-cluster sum of squares:

\[
\arg \min_S \sum_{i=1}^k \sum_{x_j \in S_i} \|X_i - \mu_i\|^2
\]

(3.2)

Where \(\mu_i\) is the mean of points in \(S_i\). The most common algorithm uses an iterative refinement technique. Due to its ubiquity, it is often called the \(k\)-means algorithm; it is also referred to as Lloyd's algorithm, particularly in the computer science community. Given an initial set of \(k\) means \(m_1^{(1)}, \ldots, m_k^{(1)}\), which may be specified randomly or by some heuristic, the algorithm proceeds by alternating between two steps (Patel and Sinha 2010). Assign each observation to the cluster with the closest mean by

\[
S_i^{(t)} = \left\{ x_j : \|x_j - m_i^{(t)}\| \leq \|x_j - m_j^{(t)}\| \text{ for all } i^* = 1, \ldots, k \right\}
\]

(3.3)

Then, calculate the new means to be the centroid of the observations in the cluster.

\[
m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j
\]

(3.4)

The \(k\)-means clustering algorithm works best in datasets that have with clusters that are equally sized. However, it performs well in data points lie in a Euclidean space and fails to perform well for more complex types of data. Despite these disadvantages, the \(k\)-means algorithm is a major unsupervised technique in the clustering analysis: It works well on many realistic data sets and
is relatively fast, easy to implement and understand. Figure 3.7 shows the computational module of the $k$-means segmentation algorithm

![Figure 3.7 The computational flow of $k$-means clustering approach](image)

### 3.3.2 Fuzzy C-Means Clustering

FCM clustering is an overlapping clustering algorithm where each point may belong to two or more clusters with different degrees of membership. The features with close similarity in an image are grouped into the same cluster. The similarity is defined by the distance of the feature vector to the cluster centers. Euclidean distance is used to measure this distance, and data will be associated with an appropriate membership value. The cluster center is updated until the difference between adjacent objective function ($I_m$), as displayed in Equation 3.5 is close to zero or practically less than a predefined small constant.
Where $M$ is an exponential weighting function that controls the fuzziness of the membership function and $c$ is some clusters. The fuzziness of the membership function used by the proposed system is 2, which is supported by the literature paper (Rose et al., 1997). If the fuzziness coefficient is increased ($> 2$), then there are no clusters formed (All pixels are grouped into one cluster). If the fuzziness coefficient is decreased ($< 2$), clusters with improper boundaries are formed (i.e. the boundary pixels in the clusters are overlapped each other). $U_{ij}$ is the degree of membership of $x_i$ in the cluster $j$, $x_i$ is the $i^{th}$ of $d$-dimensional measured data, $c_j$ is the $d$-dimension center of the cluster, and $\|*\|$ is any norm expressing the similarity between any measured feature and the center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership $U_{ij}$ and the cluster centers $c_j$ by Equations (3.6) and (3.7)

$$U_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{x_i - c_j}{x_j - c_k} \right)^{M-1}}$$ (3.6)

$$C_j = \frac{\sum_{i=1}^{M} U_{ij}^M x_i}{\sum_{i=1}^{M} U_{ij}^M}$$ (3.7)

The iteration will stop when Equation (3.8) is satisfied:

$$\max_{ij} \left\{ \left| U_{ij}^{(K+1)} - U_{ij}^{(K)} \right| \right\} < \varepsilon$$ (3.8)

Where $K$ is the iteration number. This procedure converges to a saddle point of $J_m$. The algorithm is composed of the following steps:

**Step 1:** Initialize the fuzzy partition matrix $U = [U_{ij}]_0(U(0))$ by generating random numbers in the range 0 to 1 subject to Equation (3.9):
\[ \sum_{i=1}^{M} \sum_{j=1}^{c} U_{ij} = 1 \]  

(3.9)

**Step 2:** At \( k \)-step: calculate the centers vectors \( C(k) = [c_{j}] \) with \( U(k) \) according to Equation (3.7).

**Step 3:** Update the fuzzy partition matrix \( U(k), U(K+1) \) by the new computed \( U_{ij} \).

**Step 4:** Compute the objective function according to Equation (3.6). If the difference between adjacent values of the objective function is less than termination criterion, then stop the iteration; otherwise return to step 2. Figure 3.8 shows the flow chart of FCM algorithm
3.3.3 **EM Algorithm**

The EM algorithm produces maximum likelihood estimates of parameters when there is a many-to-one mapping to the distribution governing the observation (Gupta and Chen 2011). The EM algorithm is widely used in the image segmentation field, and it produces very good results especially with a limited noise level. The image is considered as a Gaussian mixture model. Each class is represented as a Gaussian model, and the pixel intensity is assumed as an
observed value of this model. The EM is used for finding the unknown parameters of the mixture model.

A set of observed data  \( X = \{ x_i | i=1,2,\ldots,N \} \) can be modeled as to be generated from a mixture of random processes  \( X_1, X_2, \ldots, X_k \) with joint probability distribution  \( f(X_1, X_2, \ldots, X_k) \), where  \( K \) is the number of classes or distribution functions present in the mixture. It is usually assumed that these processes represent independent identically distributed random variables. Then one can write:

\[
f(X_1, X_2, \ldots, X_k) = \sum_{k=1}^{K} p_k G(x_i | \theta_k^{(t)})
\]

(3.10)

Where  \( f(x, \theta_k) \forall k=1,2,\ldots,K \) is the probability distribution function of the random variable  \( X_k \), and  \( \theta_k = \{ \mu_k, \sigma_k \} \) stands for the parameters that define the distribution  \( k \). \( \phi = \{ p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k \} \) is called the parameter vector of the mixture, where  \( p_k \) the mixing are proportions  \( (0 \leq p_k \leq 1, \forall k=1,\ldots,K, \text{and} \sum_k p_k = 1) \).

The EM algorithm consists of two major steps: an expectation step (E-step), followed by a maximization step (M-step). The expectation step is to estimate a new mapping (pixel-class membership function) on the unknown underlying variables, using the current estimate of the parameters and conditioned upon the observations. The maximization step then provides a new estimate of the parameters. Figure 3.9 shows the flow chart of EM algorithm.
EM approach divided into five sequential steps.

**Step 1:** Let \( m = 0 \) and make an initial estimate \( \theta^{(m)} \) for \( \theta \).

**Step 2:** Given the observed data \( y \) and pretending for the moment that your current guess \( \theta^{(m)} \) is correct, formulate the conditional probability distribution \( p(x|y, \theta^{(m)}) \) for the complete data \( x \).
Step 3: Using the conditional probability distribution $p(x \mid y, \theta^{(m)})$ calculated in Step 2, form the expected conditional log-likelihood, which is called the $Q$-function:

$$Q(\theta \mid \theta^{(m)}) = \int \log(p(x \mid \theta) p(x \mid y, \theta^{(m)})) dx_{(y)}$$

$$= E_{X \mid y, \theta^{(m)}}[\log p(X \mid \theta)] \tag{3.11}$$

Where the integral is over the set $X_{(y)}$, which is the closure of the set $\{x \mid p(x \mid y, \theta) > 0\}$, and we assume that $X_{(y)}$ does not depend on $\theta$. Note that $\theta$ is a free variable in (1.3), so the $Q$-function is a function of $\theta$, but also depends on your current guess $\theta^{(m)}$ implicitly through the $p(x \mid y, \theta^{(m)})$ calculated in Step 2.

Step 4: Find the $\theta$ that maximizes the $Q$-function by Equation (3.11); the result is the new estimate $\theta^{(m+1)}$.

Step 5: Let $m := m + 1$ and go back to Step 2. (The EM algorithm does not specify a stopping criterion; standard criteria are to iterate until the estimate stops changing: $\|\theta^{(m+1)} - \theta^{(m)}\| < \varepsilon$ for some $\varepsilon > 0$, or to iterate until the log-likelihood $l(\theta) = \log p(y \mid \theta)$ stops changing $|l(\theta^{(m+1)}) - l(\theta^{(m)})| < \varepsilon$ for some $\varepsilon > 0$.

The EM estimate will find a peak in the likelihood $p(y \mid \theta)$, but if the likelihood function $p(y \mid \theta)$ has multiple peaks, EM will not necessarily find the global maximum of the likelihood. In practice, it is common to start EM from multiple random initial guesses, and choose the one with the largest likelihood as the final guess for $\theta$. The traditional description of the EM algorithm consists of only two steps. The above steps 2 and 3 combined are called the E-step for expectation, and Step 4 is called the M-step for maximization:
Since the E-step is just to compute the \( Q \)-function, which is used in the M-step, EM can be summarized as just iteratively solving the M-step. When applying EM to a particular problem, this is usually the best way to think about EM because then one does not waste time computing parts of the \( Q \)-function that do not depend on \( \theta \).

### 3.3.3.1 Contrasting EM with a Simple Variant

In step 2 above, one computes the conditional distribution \( p(x \mid y, \theta(m)) \) over all possible values of \( x \), and this entire conditional distribution is taken into account in the M-step. A simple variant is to instead use only the \( m^{th} \) maximum likelihood estimate \( x(m) \) of the complete data \( x \):

\[
\text{E-like step:} \quad x^{(m)} = \arg \max_{x \in X(y)} p(x \mid y, \theta^{(m)}) \quad (3.12)
\]

\[
\text{M-like step:} \quad \theta^{(m+1)} = \arg \max_{\theta \in \Omega} p(x^{(m)} \mid \theta) \quad (3.13)
\]

EM clustering differs from \( k \)-means clustering in that at each iteration there is no need to choose a single \( x(m) \), that is, one does not force each observed point \( y_i \) to belong to only one cluster. Instead, each observed point \( y_i \) is probabilistically assigned to the \( k \) clusters by estimating \( p(x \mid y, \theta(m)) \).
3.3.3.2  Using a Prior with EM (MAP-EM)

Sometimes EM algorithm can fail due to singularities of the log likelihood function for example while learning a GMM with 10 components, it may decide that the most likely solution is for one of the Gaussians to only have one data point assigned to it, with the bad result that the Gaussian is estimated as having zero covariance. A straightforward solution to such degeneracies is to take into account or impose some prior information on the solution for \( \theta \). One approach would be to restrict the set of possible \( \theta \). Such a restriction is equivalent to putting a uniform prior probability over the restricted set. More generally, one can impose any prior \( p(\theta) \) and then modify EM to maximize the posterior rather than the likelihood:

\[
\hat{\theta}_{MAP} = \arg \max_{\theta \in \Omega} \log p(y \mid \theta)= \arg \max_{\theta \in \Omega} (\log p(y \mid \theta) + \log p(\theta)) \tag{3.14}
\]

The EM algorithm is easily extended to maximum a posteriori (MAP) estimation by modifying the M-step:

**E-step:** Given the estimate from the previous iteration \( \theta^{(m)} \), compute as a function of \( \theta \in \Omega \) the conditional expectation

\[
Q(\theta \mid \theta^{(m)}) = E_{X \mid y, \theta^{(m)}}[\log p(X \mid \theta)]
\]

**M-step:** Maximize \( Q(\theta \mid \theta^{(m)}) + \log p(\theta) \) over \( \theta \in \Omega \) to find

\[
\theta^{(m+1)} = \arg \max_{\theta \in \Omega} (Q(\theta \mid \theta^{(m)}) + \log p(\theta))
\]