Pattern Recognition

Midterm Project

Submitted by:

1. **Ahmed Fawzi**
2. **Hendwan Abozide**
3. **Rokaia Mohamed**

Problem Definition and Importance

The problem is that we want to view specific details from WBC, from two datasets (ex. Nucleus, Cytoplasm), and ignore the rest of the image.

We need to segment the image for better visualization of the cells. There are **3** different ways we will mention in this report that could segment the image.

Methods and Algorithms

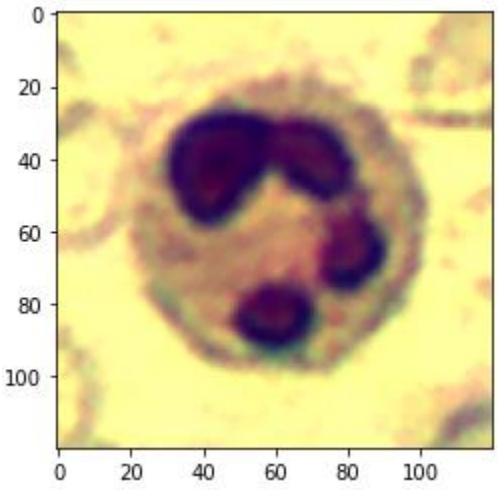
1. C-means clustering

Steps to implement it:

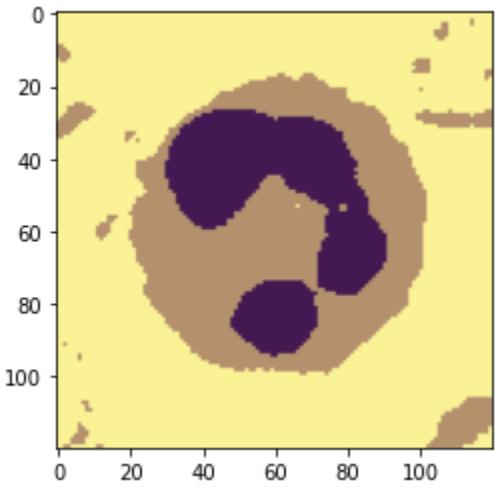
1. Number of clusters is entered by user (K).
2. Centroids of each cluster are calculated.
3. Euclidean distance is calculated between each pixel and the cluster, and the pixel is assignment to the cluster with the minimum distance.
4. Adjust the value of centroids by iterating until convergence (old value of convergence equal the updated value); when no change to the centroids occurs.

Output Example

Original Image



Output after C-Means Clustering



Code Example

Calculating Centroids:

```
if ((int(old_cent[0]) - 1) <= cent[0] <= (int(old_cent[0]) + 1)) and
((int(old_cent[1]) - 1) <= cent[1] <= (int(old_cent[1]) + 1)) and
((int(old_cent[2]) - 1) <= cent[2] <= (int(old_cent[2]) + 1)):
```

Where **old_cent** is the old centroids & **cent** is the new one.

Assigning Centroids to the cluster:

```
clusters[minIndex].append(p)
```

Where **p** is the pixel with the minimum distance from the cluster.

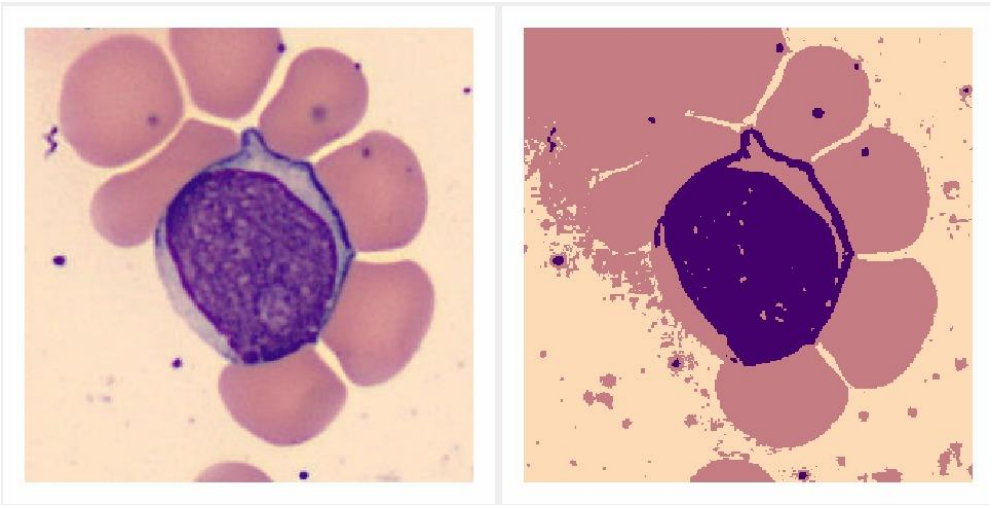
2. Multi-Class Neural Network

Steps to implement it:

1. Initialize random weights array and bias arrays
2. We chose the activation function to be **ReLU**
3. Calculating the output probability of layer using **softmax**
4. Then calculating the loss function of each softmax value using **cross_entropy**
5. Predict the labels
6. Plot output image same way as in SVM

Output Example

Original Image **Output after SVM**



Output image is similar to the output of SVM, but the output is plotted much faster than SVM.

Code Example

ReLU Actvation Function

```
np.maximum(data,0)
```

Getting output of layer and calculating softmax

```
input_layer = np.dot(training_data, self.layer1_weights_array)
hidden_layer = relu_activation(input_layer + self.layer1_biases_array)
output_layer = np.dot(hidden_layer, self.layer2_weights_array)
self.layer2_biases_array
output_probs = softmax(output_layer)
```

Predicting labels

```
input_layer = np.dot(list_1, layer1_weights_array)
hidden_layer = relu_activation(input_layer + layer1_biases_array)
scores = np.dot(hidden_layer, layer2_weights_array) + layer2_biases_array
probs = softmax(scores)
```

where prob is the output y_pred

3. SVM

Steps to implement it:

1. Take snippets from the images (Background, Inner and outer) images from the two datasets and put each group of images in different folders.

2. We generate labels from each pixel read from each Background, Inner & outer. Background has value 0, Inner has value 1 and Outer has value 2.

3. Generating the X train array values (pixel_array) in each in pixel read done on Background, Inner and Outer. Divide pixel value by 255 for normalization.

4. Using **svclassifier** fit the training data with the labels and output the **y_pred**

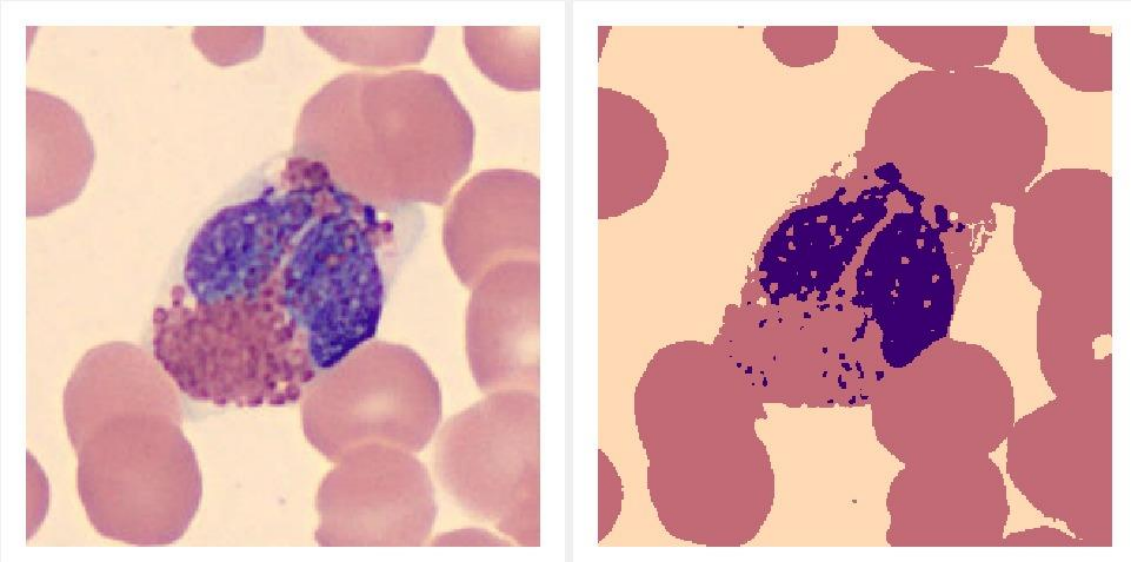
5. Check the y_pred values,
   If y_pred=0 take the corresponding pixel in test image and append it in the background list.
   If y_pred=1 take the corresponding pixel in test image and append it in the inner_cell list.
   If y_pred=2 take the corresponding pixel in test image and append it in the outer_cell list.

6. Then get the mean of these lists and add them to new 3 arrays and Draw the output image.

Note: It may take time to run the SVM, so please be patient ☺

Output Example

Original Image **Output after SVM**



Code Example

Generating the **pixel_list (Train_x)** data and the labels and repeating this for **inner & outer**

```
for i in range(len(backa)):
    for i in img:
        pixel_list.append(i/255)
        Labels.append(0)
```

Checking the **y_pred** values and appending in the background, inner and outer list.

```
if(y_pred[i] ==0):
    Back_list.append(list_1[i])
if(y_pred[i] ==1):
    inner_list.append(list_1[i])
if(y_pred[i] ==2):
    outer_list.append(list_1[i])
```