Untitled

May 8, 2019

In [ ]: Hausaufgaben von Wajid Ghafoor und Benjamin Ostendorf

In [462]:
```
from PIL import Image, ImageColor
import numpy as np
from PIL import ImageFilter
import math as m
```

In [463]:
```
# Show one image
im = Image.open("positives/p01.png")
im.show()
```

In [464]:
```
def extrema_red(image):
    return image.getextrema()[0][0]
def extrema_green(image):
    return image.getextrema()[1][0]
def variance_red(image):
    return np.var(image.split()[0])
def variance_green(image):
    return np.var(image.split()[1])
def mean_blue(image):
    return np.mean(image.split()[2])
```

In [465]:
```
def create_feature(image):
    er = extrema_red(image)
    eg = extrema_green(image)
    vr = variance_red(image)
    vg = variance_green(image)
    mb = mean_blue(image)
    return [er,eg,vr,vg,mb]
```

In [467]:
```
def create_feature_positives():
    positives = []
    for i in range(1,31):
        if i < 10:
            im_pos = Image.open("positives/p0" + str(i) + ".png")
            er = extrema_red(im_pos)
            eg = extrema_green(im_pos)
```

```
                vr = variance_red(im_pos)
                vg = variance_green(im_pos)
                mb = mean_blue(im_pos)
                positives.append([er,eg,vr,vg,mb])
            else:
                im_pos = Image.open("positives/p" + str(i) + ".png")
                er = extrema_red(im_pos)
                eg = extrema_green(im_pos)
                vr = variance_red(im_pos)
                vg = variance_green(im_pos)
                mb = mean_blue(im_pos)
                positives.append([er,eg,vr,vg,mb])
        return positives

    def create_feature_negatives():
        negatives = []
        for i in range(1,31):
            if i < 10:
                im_neg = Image.open("negatives/n0" + str(i) + ".png")
                er = extrema_red(im_neg)
                eg = extrema_green(im_neg)
                vr = variance_red(im_neg)
                vg = variance_green(im_neg)
                mb = mean_blue(im_neg)
                negatives.append([er,eg,vr,vg,mb])
            else:
                im_neg = Image.open("negatives/n" + str(i) + ".png")
                er = extrema_red(im_neg)
                eg = extrema_green(im_neg)
                vr = variance_red(im_neg)
                vg = variance_green(im_neg)
                mb = mean_blue(im_neg)
                negatives.append([er,eg,vr,vg,mb])
        return negatives
```

```
In [468]: positive_features = create_feature_positives()
          negative_features = create_feature_negatives()
```

```
In [469]: def phi(positive_features, negative_features):
              return float(len(positive_features)) / float((len(negative_features)+len(positive
```

```
In [470]: def mue(features):
              sum_of_features = reduce(lambda a,b: [x+y for x,y in zip(a,b)], features)
              return list(map(lambda x: float(x/len(features)), sum_of_features))
```

```
In [471]: def covariance_matrix(positive_features,negative_features):
              mue0 = mue(negative_features)
              number_of_features = len(negative_features[0])
              neg_mat_list = list(map(lambda x: np.matmul(np.reshape([a-b for a,b in zip(x, mue

              mue1 = mue(positive_features)
              pos_mat_list = list(map(lambda x: np.matmul(np.reshape([a-b for a,b in zip(x, mue

              mat_list = np.append(neg_mat_list, pos_mat_list, axis=0)
              #print(mat_list)
              cov_mat = mat_list[0]
              for i in range(1,len(mat_list)):
                  cov_mat = np.add(cov_mat,mat_list[i])

              return np.multiply(cov_mat,float(1.0/float(len(mat_list))))

In [483]: def predict(image,positive_features,negative_features):
              n = len(positive_features)+ len(negative_features)
              phi_1 = phi(positive_features,negative_features)
              phi_0 = phi(negative_features, positive_features)
              #print(phi_1)
              #print(phi_0)
              mue0 = mue(negative_features)
              mue1 = mue(positive_features)
              cov_mat = covariance_matrix(positive_features, negative_features)
              image_features = create_feature(image)
              f_len = len(image_features)
              function_prefix = 1.0/((2.0*m.pi)**float(n/2.0)*np.linalg.det(cov_mat)**float(1.0
              in_cov_mat = np.linalg.inv(cov_mat)
              f_diff_0 = np.reshape([x+y for x,y in zip(image_features,mue0)], (1, f_len))

              mat0_1 = np.multiply(-0.5, f_diff_0)
              mat0_2 = np.matmul(mat0_1, in_cov_mat)
             # print(np.matmul(mat0_2, np.reshape(f_diff_0, (f_len, 1))))
              px_0 = function_prefix*m.e**(np.matmul(mat0_2, np.reshape(f_diff_0, (f_len, 1))))

              f_diff_1 = np.reshape([x+y for x,y in zip(image_features,mue1)], (1, f_len))

              mat1_1 = np.multiply(-0.5, f_diff_1)
              mat1_2 = np.matmul(mat1_1, in_cov_mat)
             # print(np.matmul(mat1_2, np.reshape(f_diff_1, (f_len, 1))))
              px_1 = function_prefix*m.e**(np.matmul(mat1_2, np.reshape(f_diff_1, (f_len, 1))))

              #print(phi_1, mue0, mue1, cov_mat)
              if phi_0 * px_0 > phi_1 * px_1:
                  return "negativ, u are not infected"
              else:
                  return "positive, chargas Parasite found!!!"
```

```
In [484]: for i in range(10,30):
              print(predict(Image.open("positives/p" + str(i) + ".png"), positive_features,nega
```

```
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
```

```
In [485]: for i in range(10,30):
              print(predict(Image.open("negatives/n" + str(i) + ".png"), positive_features,nega
```

```
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
negativ, u are not infected
```