**Problem 1**

Question 1

All we need to prove is that there exists a group of 3 points shattered by , but every group of 4 points cannot be shattered by .

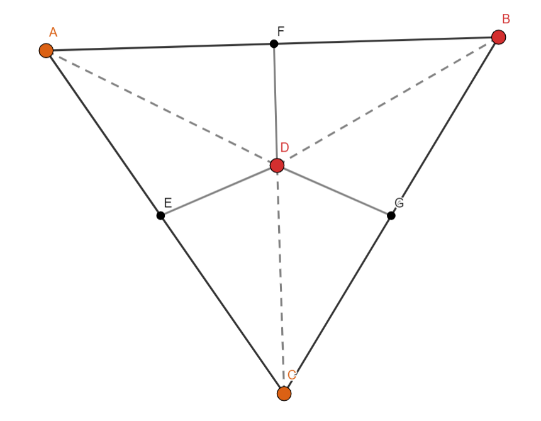


Now let's look at a set of 4 points, . There could be 2 cases.

1. 3 colinear points, therefore the middle one cannot be assigned – with the sides being labeled +
2. There is no middle point and no colinear points, and therefore by looking at the two farthest facing pair, a circle that encloses them whose center must be at some point between them which must include a third point, therefore we cannot assign + only to them.
3. There is a middle point, and thus by looking at circles built around the edges of the triangle we will prove they cover the whole triangle, meaning that the middle point cannot be assigned – with some 2 points being assigned +. We will prove that every point in triangle must be in at least one circle of the formerly mentioned circles. We will prove that:

Assume by contradiction that , then we can conclude that:

And if we sum over all sub-triangles, we get that , which is incorrect. Therefore, we've proven that no hypothesis could shatter a group of this kind.



Note: diagrams are disgusting sorry 😊

Question 2

Question 3

The problem is now PAC learnable, therefore, according to the fundamental theorem of statistical learning:

**Problem 4**

Section a

We want to find local minimum, which can be done by finding points where function gradient of is 0.

Section b

I added two lines of code which initialize a Kmeans object, and fits it to the data.

k = 2

clust = Kmeans(k)

clust.fit(data)

I also added a function that reassigns the centroids after slef.labels has been updated by the class

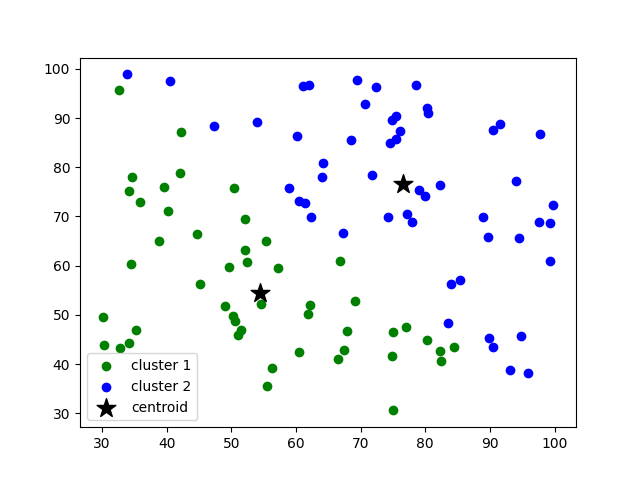
    def reassign\_centroids(self, X, labels):

        centroids = np.zeros((self.n\_clusters, X.shape[1]))

        for i in range(self.n\_clusters):

            centroids[i] = np.mean(X[labels==i])

        return centroids

here are the results:

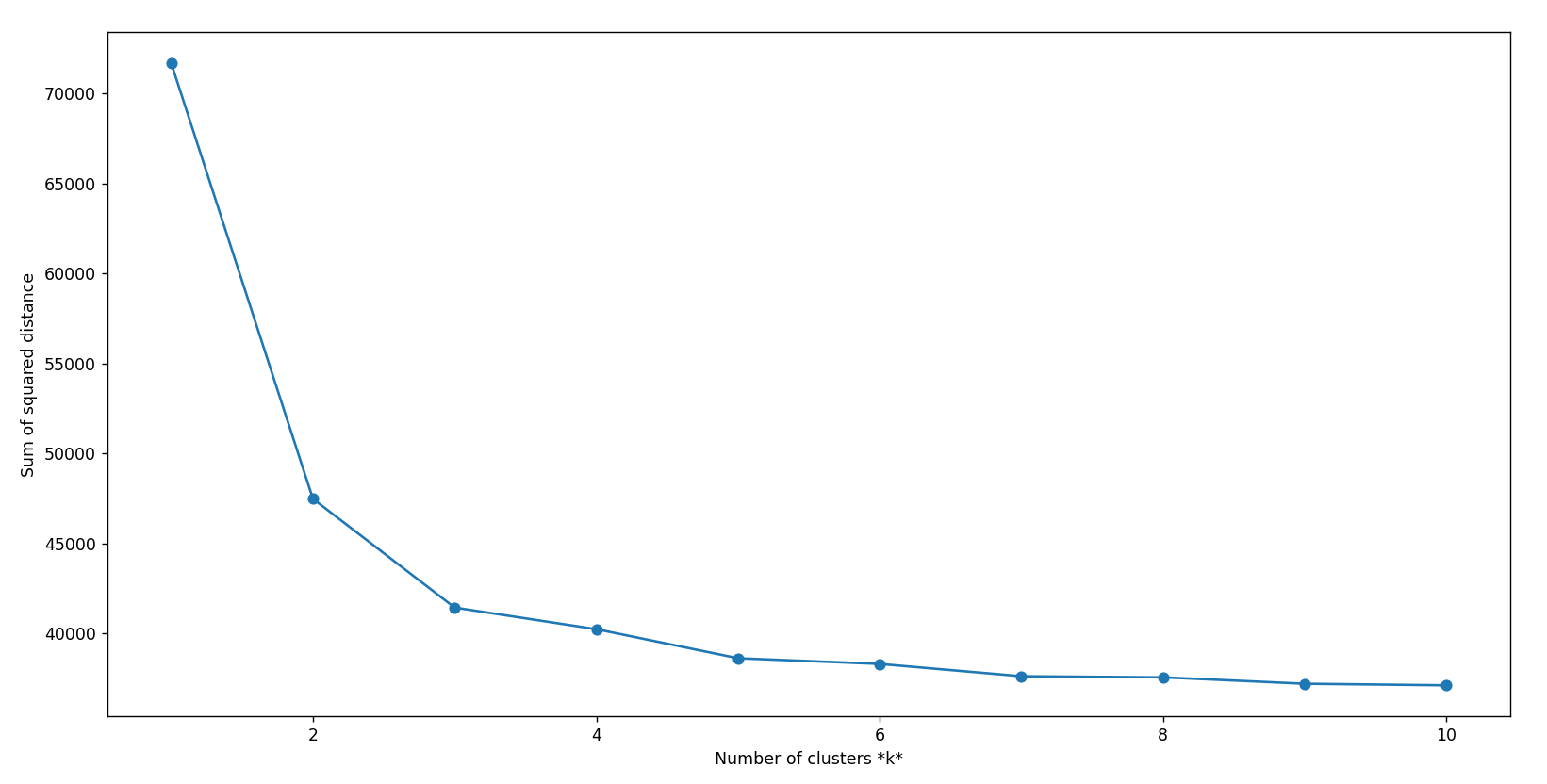
and using the elbow function we plotted the inertia of the algorithm as a function of k.

for k in list\_k:

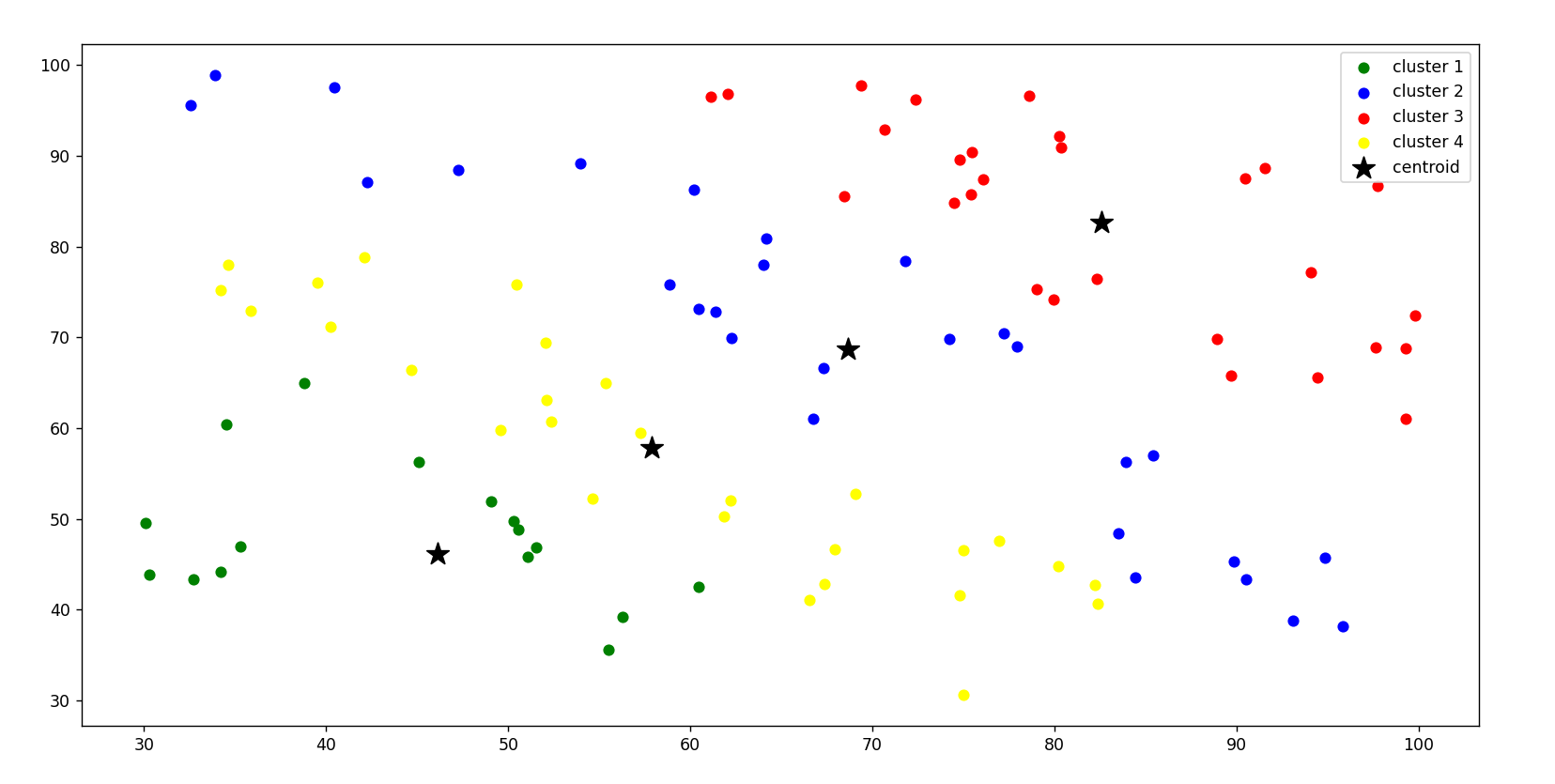
    clust = Kmeans(k)

    clust.fit(data)

    sse.append(clust.compute\_sse(data,clust.labels,clust.centroids))



And therefore the best choice is k=4 or k=3 (somewhat between overfitting and underfitting, high bias and high variance, which is right after the drastic change). Here are the results of k=4.



Section D

I used sickit-learn class for implementing k-means, and found clustering for 20 colors. Then, I built a function that takes the k-means object and the picture, and for each pixel, maps the compressed color. Then we plotted both pictures. Here is the code and the output.

# returns compresseed image from image X by cluster clust

def compress(X,clust):

    for i in range(20):

        X[clust.labels\_==i] = clust.cluster\_centers\_[i]

    return X



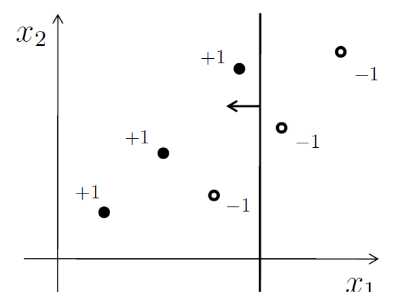
**Problem 5**

1. Initializing we get that , and
2. For misclassified points, .

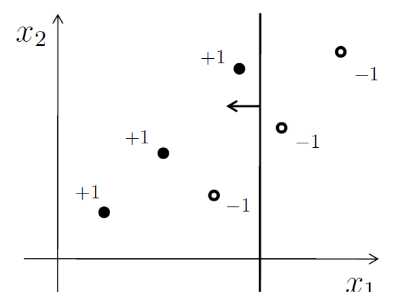
For well classified points,

To normalize we divide by , and get

1. We lose one positive point for the expensive negative point, and get that:



1. , and since is a monotonically decreasing function, .
2. We proved that if we choose the same weak classifier with , we will get . This can be proven to be correct with the blue weak classifier, in which , which is not the case for green classifier.
3. We draw all of them and take the weighted sum sign:



Train accuracy rate is 100% since all samples are eventually classified correctly.