Roi Herzig ID: 300360310 EMAIL:roeiherzig@mail.tau.ac.il

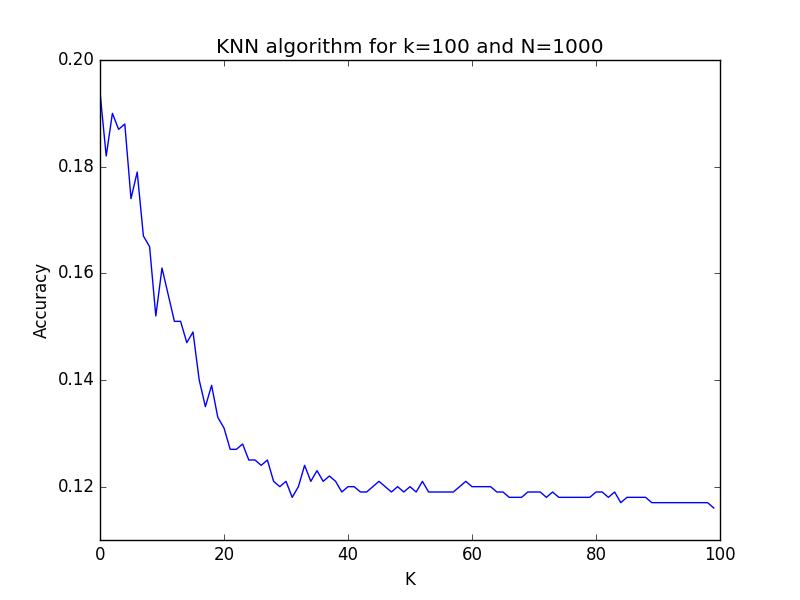
Moshe Raboh ID:300611878 EMAIL:shikorab@gmail.com

**Introduction To Machine Learning – EX 1**

1. **A:**See function "knn" in file "NearestNeighbor.py"

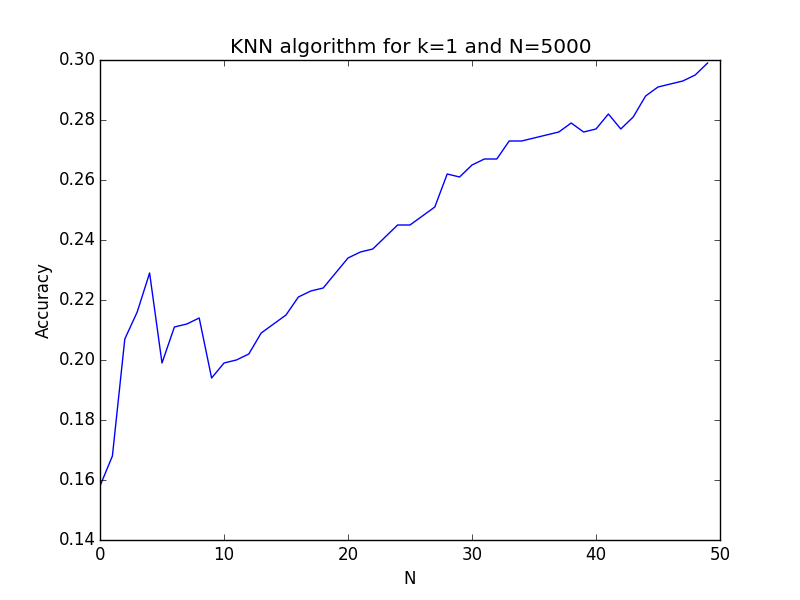
**B:**  
The accuracy of prediction of KNN with k=10 over 1000 training images (n=1000) is 0.157.  
We would expect from a completely random predicate accuracy of prediction of 0.1 (we have 10 different labels).

**C:**See function "get\_accuracy\_of\_k" in file "NearestNeighbor.py"  
Image file: " graph\_N\_fixed.png"

The best k is 1- (about the same accuracy).  
As we can see in the graph:  
for small Ks (1 to 3) we get the highest accuracy. We can see that the accuracy descend and approaches 0.1 when k growth.  
0.1 is the accuracy is the accuracy we would expect from a completely random predicator.

We would expect for such a behavior, since:

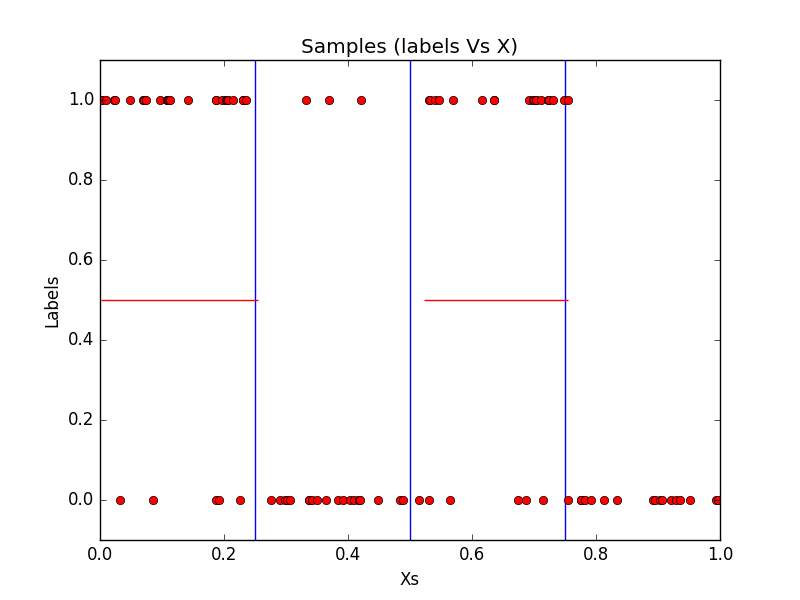
1. Euclidian distance is not accurate (however better than a random predicator)
2. The influence of each of the k nearest neighbors is equal (meaning that the nearest neighbors and the k nearest neighbor will equally influence the result).  
   Since that the training set is sparse (only 1000 images).

**D:**See function "get\_accuracy\_of\_k" in file "NearestNeighbor.py"Image file: " graph\_K\_fixed.png" ****According to the graph we can see that we get better accuracy with bigger training size.   
We ran it with k=1 (the best results from above).  
This is the expected results, since it will be possible to find better (closer) single nearest neighbor which better reflect (statistically of course) the true label.

**2.**

**A:**Function "part\_a" in file "UnionOfInterval.py"

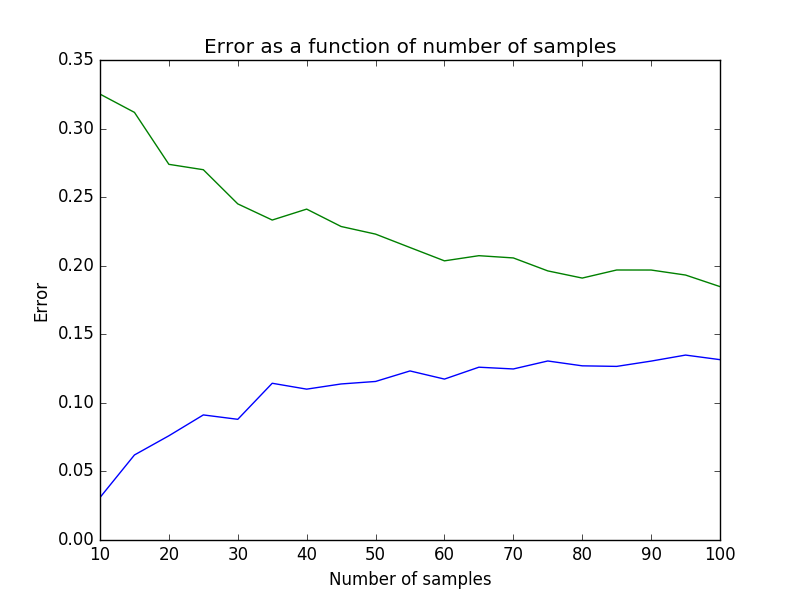
The image file name: "partA.png"

****

The red lines are the intervals (the result of function "find\_best\_intervals" with k = 2.  
The blue lines are vertical lines is x=0.25, 0.5, 1.

**B:**

Given the distribution, the hypothesis with the smallest error will be hypothesis of 2 intervals ((0, 0.25), (0.5, 0.75)).  
The error is 15% (2\*0.25\*0.2 + 2\*0.25\*0.1 = 0.15)

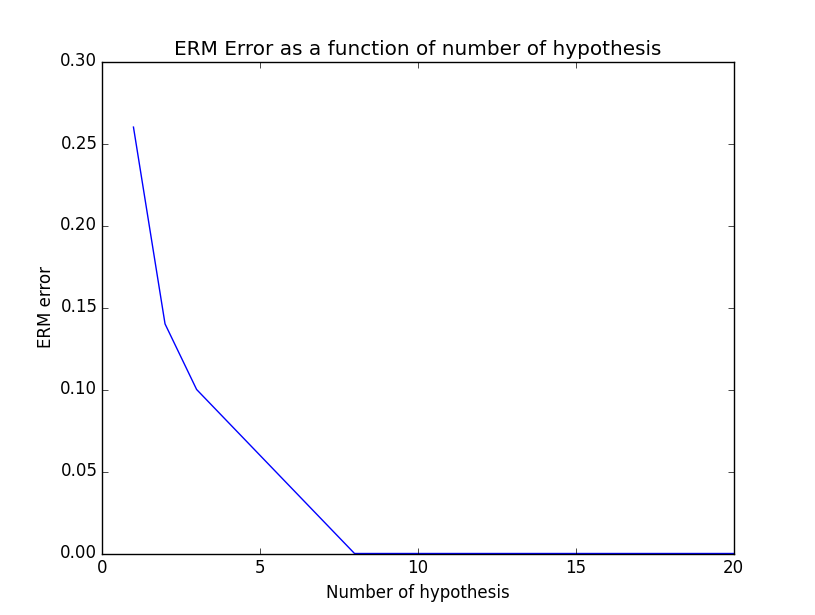
**C:**True error function: "calc\_true\_error" in file"UnionOfInterval.py".  
The experiment function: "part\_c" in file "UnionOfInterval.py".  
The image file name is "partC.png" ****The green line is the true error as a function of m.  
The blue line is the empirical error as a function of m.

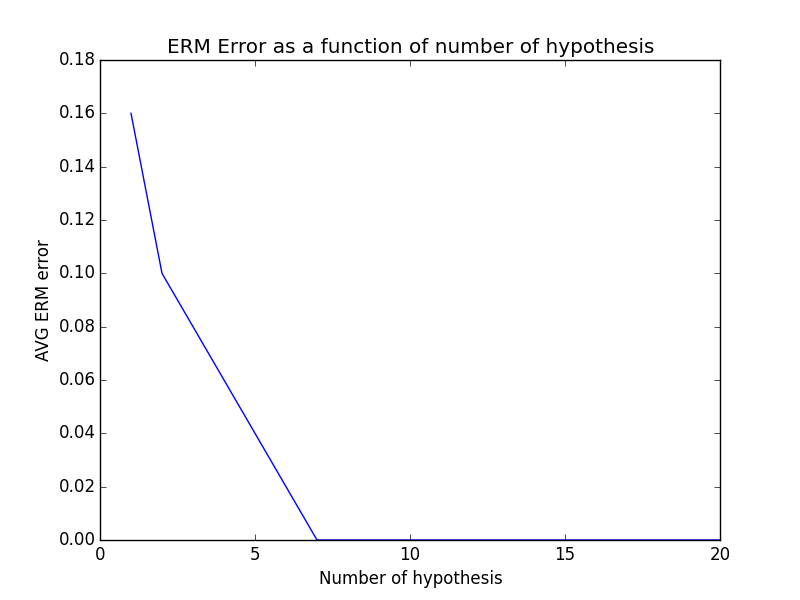
As we can see:

The empirical error increasing when m grows. When m is small, there are small amount of samples, sparse, so it is easier to find intervals with better results (smaller error) compared to the true hypothesis (on these samples only of course)**.** In such a case the ERM overfits.

The true error decreasing when m is grows. The true hypothesis best fit to the theoretical distribution. The samples will better reflect the theoretical distribution when m is big. **D:**

Function name "part\_d\_and\_e(t = 1, file\_name="partD.png")" in file "UnionOfInterval.py".  
Image file is "partD.png"

The error decreasing when k growth. When single interval allows to reduce the error with at least one (when the error != 0 of course).  
K\* will is any k bigger then 8. Of course K\* is not a good choice since it overfits the samples.  


E:  
Function name "part\_d\_and\_e(t = 100, file\_name="partE.png")" in file "UnionOfInterval.py".  
Image file is "partE.png"  


F:   
function name: part\_f() in file in file "UnionOfInterval.py".

Given a new data set with m=50, we would perform cross validation in the following way:  
- Get the best hypothesis per k from section D.  
- Compare the hypothesis labels with the true labels of the new dataset (validation data set).  
- Pick the best hypothesis (the lowest number of errors over the new data set).

This cross validation will allow us to avoid from model which overfit the original sample from section D.  
Indeed with relatively small number of samples we were able to find a hypothesis which is very close to the theoretical best hypothesis. (same number of intervals and overlapping intervals).  
intervals = (0.005, 0.24), (0.49, 0.72)

Error = 0.14

Note:  
Dividing the samples to a bigger number of training samples vs validation samples might produce better results.