Introduction to Machine Learning - Exercise 1

Due Date: April 15th, 2019 (there will be no extensions!)

Yossi Adi, Felix Kreuk and Joseph Keshet

April 4, 2019

1 ERM

1.1

As mentioned in the class, a learning algorithm receives as input a training set S sampled from an unknown distribution $\mathcal{D}$ and labeled by some target function f. Since the learner does not know what $\mathcal{D}$ and f are, we use a training set of examples, which acts as a snapshot of the world that is available to the learner. In ERM we would like to find a solution that works well on that data.

An aligned circle classifier in the plane is a classifier that assigns the value 1 to a point if and only if it is inside a certain circle. Formally, given a point (c_1, c_2) and a radius r, define the classifier $h(c_1, c_2, r)$ by,

$$h(x; c_1, c_2, r) = \left\{ \begin{array}{ll} 1 & \text{if } \sqrt{(c_1 - x_1)^2 + (c_1 - x_2)^2} \leq r \\ 0 & \text{otherwise} \end{array} \right.$$

Let A be the algorithm that returns the smallest circle enclosing all positive examples in the training set. Explain why A is **NOT** an ERM.
Bonus: Briefly describe for what minimal geometric shape the above claim holds.

1.2

Let $\mathcal{H}$ be the hypothesis space of binary classifiers over a domain $\mathcal{X}$. Let $\mathcal{D}$ be an unknown distribution over $\mathcal{X}$, and let f be the target hypothesis in $\mathcal{H}$. Denote $h \in \mathcal{H}$.
Let us define the *true error* of h as,

$$L_{\mathcal{D}}(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$

Let us define the *empirical error* of h over the training set S as,

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{[h(x) \neq f(x)]}$$

where m is the number of training examples.
Show that the expected value of $L_S(h)$ over the choice of S equals $L_{\mathcal{D}}(h)$, namely,

$$\mathbb{E}_{S \sim \mathcal{D}}\big[L_s(h)\big] = L_{\mathcal{D}}(h)$$

2 Image Compression

In this part of the exercise we will use the k-means algorithm for image compression, i.e. you should implement the k-means algorithm on the **image pixels** and then replace each pixel by its centroid. You should implement the k-means algorithm as described in class (slide no. 20 in recitation 1 presentation).
You will train your algorithm and report results using the picture in Figure 1. You will be provided with a python script, named `load.py` for reading, normalizing and reshaping the image so it will be ready for training.

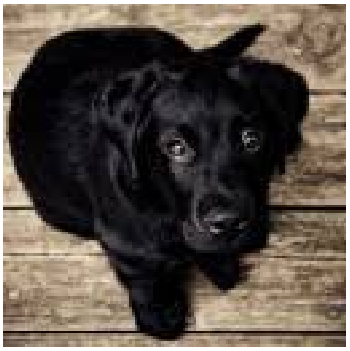

Figure 1: Dog

You should get something similar results to the following,

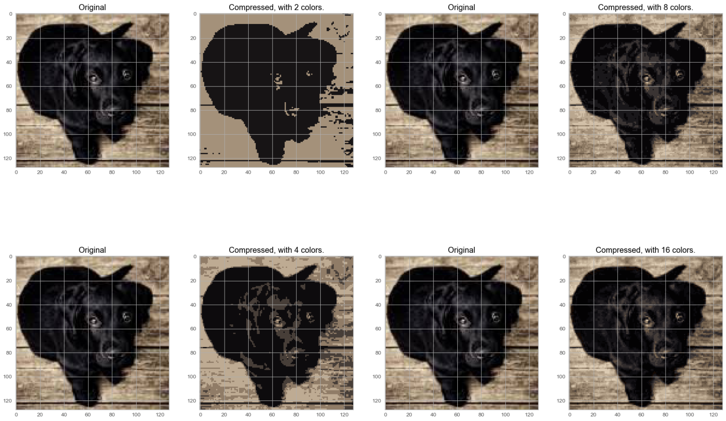

Figure 2: Results

Reproducibility. Originally, the initial centroids in k-means are randomly generated. For reproducibility purposes, we provided you with a python script named, `init_centroids.py` for centroid initialization, you should use it! Please note that given these pre-defined values, your sequence of centroid updates should be determinstic and not random in any way.

Your code should run for 10 iterations for each value of k. The output should consist of your centroids after each centroid update step as follows:

```
k=2:
iter 0: [c_11, c_12, c_13], [c_21, c_22, c_23]
iter 1: [c_11', c_12', c_13'], [c_21', c_22', c_23']
...
k=4:
iter 0: [c_11, c_12, c_13], [c_21, c_22, c_23], [c_31, c_32, c_33], [c_41, c_42, c_43]
iter 1: [c_11', c_12', c_13'], [c_21', c_22', c_23'], [c_31', c_32', c_33'], [c_41', c_42', c_43']
...
```

Where the centroids at iter 0 are the initial centroids, and the centroids at iter i are the centroids after i update steps, overall $0 \leq i \leq 10$. Centroid coordinates should be printed at a precision of 2 digits after the decimal point after the *floor* operator(e.g., floor(2.45673) = 2.45).

3 What to submit?

You should submit the following files:

- A `txt` file, named `details.txt` with your name and ID.
- A `PDF` file with your answers to 1.1 and 1.2.
- Python 3.6 code for question 2. Your main function should reside in a file called `ex1.py`. The main function should output centroids for $k = 2, 4, 8, 16$ as explined above.
- A `PDF` report with all the implementation details, (number of iterations, how you chose you centroids, did you run multiple run each time?, etc). The `PDF` report should also include the following plots: (1) The average loss value (i.e. the distance between each point to its closest centroid) as a function of the iterations for $k = 2, 4, 8, 16$; (2) The final (iter 10) compressed image for $k = 2, 4, 8, 16$.
- Part of your grade will consist of automatic checks using the `Submit` system. Make sure your output matches the expected output.
- **Note**: your code should load an image named `dog.jpeg` located in the same folder as your main script. Make sure you use relative paths when loading the image.