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,

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

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.

Note: We rely on the realizability assumption.

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 Sound Compression

Guidelines

1. You are not allowed to use external packages other than numpy and scipy.
2. In order to submit your solution please upload your files to `Submit` and check your inbox for the feedback mail.
3. Technical questions about this exercise should be asked at the course' piazza.
4. Private/Personal issues regarding the deadline should be directed to **Yosi shrem**.

In this part of the exercise we will use the k-means algorithm for sound compression, i.e. you should implement the k-means algorithm on the **amplitude values** and then replace each value by its centroid.
You should implement the k-means algorithm as described in class (slide no. 15 in recitation 2 presentation). You will train your algorithm and report results using the `sample.wav` as shown in Figure 1. For visualization, you can use Praat (`http://www.fon.hum.uva.nl/praat/`) (Non-mandatory for this exercise)

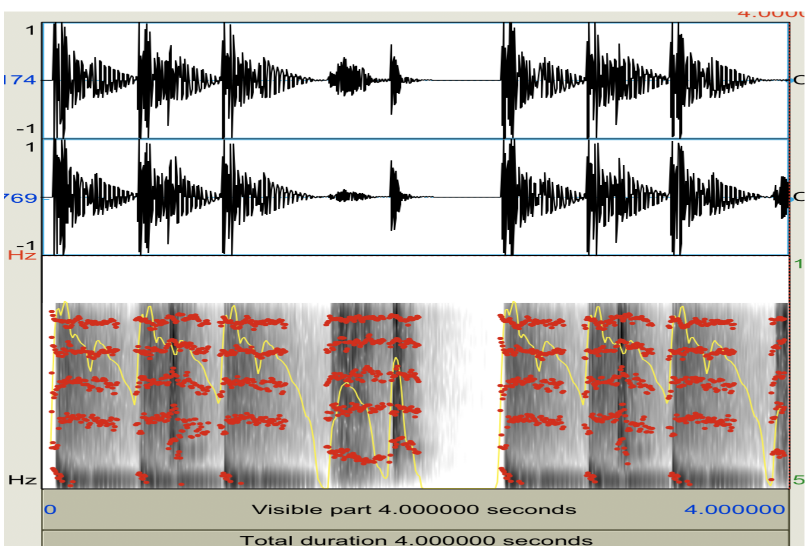

Figure 1: sample.wav

The centroids initialization will be provided to you in a text file which will be received as an argument to your program. The run command to your program should be:

```
$ python ex_1.py <wav_file_path> <centroinds_init_path>
For example:
$ python ex_1.py sample.wav cents1.txt
```

```
sample,centroids = sys.argv[1],sys.argv[2]
fs, y = scipy.io.wavfile.read(sample) #reading
x=np.array(y.copy())
centroids=np.loadtxt(centroids)

#your code goes here#
...
...
scipy.io.wavfile.write("compressed.wav", fs, np.array(new_values, dtype=np.int16))#saving
```

The following snippet contains commands for reading and writing sound files as well as reading the centroids initialization:

When displaying your compressed file you should get similar results to the following,

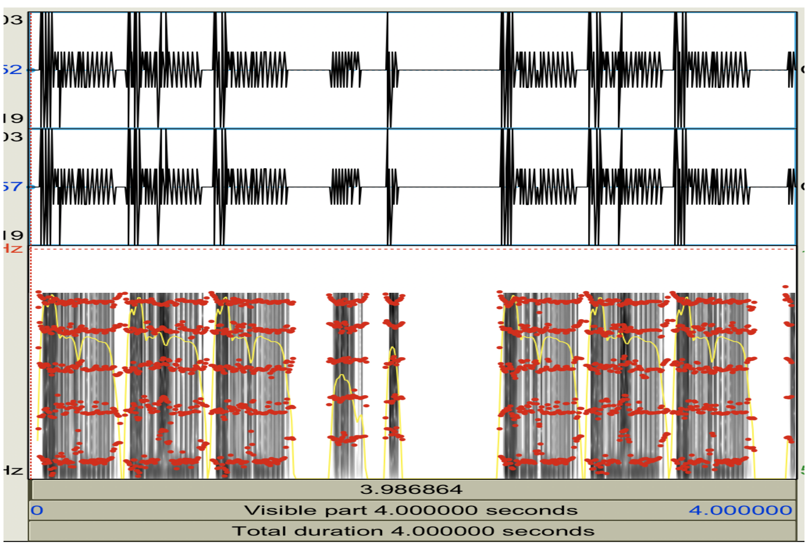

Figure 2: K=5

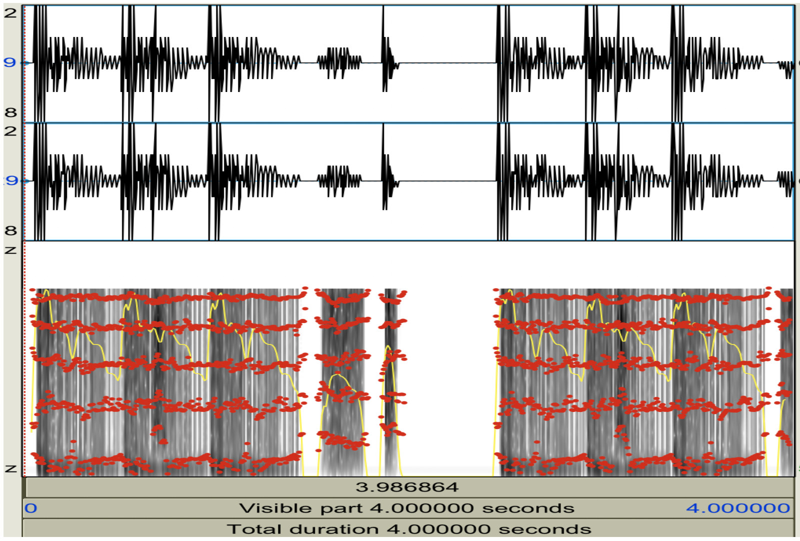

Figure 3: K=10

Reproducibility. Originally, the initial centroids in k-means are randomly generated. For reproducible purposes we provided you with the centroids initialization. Moreover, In case when 2 centroids are evenly close to a certain point, the one with the lower index "wins". To evaluate your program outputs, we provided you with 2 examples - `cents1.txt` and `cents3.txt` for centroid initialization, and with `output1.txt` and `output3.txt` the requested outputs. Please note that given these pre-defined values, your sequence of centroid updates should be deterministic and not random in any way.
Note: Your code should run for 30 iterations or until convergence. Your program should **create** a file named `output.txt`, consisting of your centroids after each centroid update step as follows:

For example, when using `cents3.txt` the requested out should be:

```
[iter 0]:[-4189. -4171.],[ 4. -2.],[3. 2.],[4. 3.],[4543. 4517.]
[iter 1]:[-7645. -7603.],[-494. -718.],[-243.   24.],[780. 802.],[7851. 7858.]

...
[iter 18]:[-24823. -24965.],[-7598. -7574.],[ 10. -11.],[7645. 7638.],[24409. 24914.]
[iter 19]:[-24823. -24965.],[-7598. -7574.],[ 10. -11.],[7645. 7638.],[24409. 24914.]

The full requested output is at output3.txt
use the following line to match your output to the requested format:
   f"[iter {iter}]:{','.join([str(i) for i in new_z])}"
```

As you can see the algorithm converged and stopped(same centroids). **For consistency, after each centroids update, use the built-in** *round*() **function on each dimension**. **We define convergence when the centroids don't update.**

The `sample.wav` was taken from here : `https://www.youtube.com/watch?v=7vQ83lz0528`

(Lucille Crew - Something).
Lastly, your code can not run on the u2 servers, to overcome formatting issues, we ask you to run your code using the `cents2.txt` as the centroids initialization, and then manually change the filename from `output.txt` to `output2.txt`. Once submitted you will get a feedback email describing whether the expected output matches the requested format- check it and correct if needed. Part of your grade will consist of automatic checks - follow the format guidelines.

3 What to submit?

You should submit the following files:

- A `txt` file, named `details.txt` with your name and ID.
- A `PDF` file named `ex_1.pdf` 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 `ex_1.py`. The main function writes the centroids to `output.txt` as explained above.
- A txt file named `output2.txt` - Your program's output when using the `cents2.txt` for initialization.(i.e. run your code using the `cents2.txt` and then manually change the filename from `output.txt` to `output2.txt`)
- A `PDF` report including the following plots: The average loss/cost value as a function of the iterations(you can stop at 10) for $k = 2, 4, 8, 16$ (4 plots in total). Explain shortly about your centroids initialization process.

  Overall : `ex_1.py, ex_1.pdf, details.txt, output2.txt` and `report.pdf`

  Good Luck!