Project 4 on Mathematics in AI

Subject: Within Distance

Name: Hesam Mousavi

Student number: 9931155

How I store the dataset

I created a module called 'Dataset' which contains everything about our dataset (sample, label, number of sample, number of feature, and representor)

In [1]:

```python
from copy import deepcopy
import numpy as np
from dataset import Dataset
import numpy_indexed as npi

dataset = Dataset('dataset/iris.data')

norm_set = np.array((
    [2, 0],
    [np.inf, 0],
    [2, 1],
    [np.inf, 1],
    [1, 2],
    [np.inf, 2],
    [1, np.inf],
    [2, np.inf]
))
```

Find $e^{d,d'}(X, c)$

In [2]:

```python
def find_err(dataset: Dataset, d_norm: int, dp_norm: int):
    err_from_rep = [
        np.linalg.norm(sample - dataset.representor, d_norm)
        for sample in dataset.sample]
    return np.linalg.norm(err_from_rep, dp_norm)
```

Fisrt idea

As a first step, let's use an alternate search to get a good approximation of the representative on our dataset with our set of norms

d	d'
2	0
∞	0
2	1
∞	1
1	2
∞	2
1	∞
2	∞

For that, I use step-decay, which after a few steps without any improvement, it will half the step size. And I'll return the best representative that we've seen if the step size is smaller than epsilon

In [3]:

```
def find_rep_with_AS(dataset: Dataset, d_norm: int, dp_norm: int):
    step_size, no_improve = 1, 0
    eps, no_improve_threshold = 1e-04, dataset.number_of_feature ** 3
    random_sample = np.random.randint(0, dataset.number_of_sample)
    dataset.representor = dataset.sample[random_sample]
    best_err, best_rep = \
        find_err(dataset, d_norm, dp_norm), dataset.representor

    while(step_size > eps):
        random_feature = np.random.randint(0, dataset.number_of_feature)
        dataset.representor[random_feature] += step_size
        this_err = find_err(dataset, d_norm, dp_norm)

        if(this_err < best_err):
            best_err = this_err
            no_improve = 0
            best_rep = deepcopy(dataset.representor)
            continue
        dataset.representor[random_feature] -= step_size

        dataset.representor[random_feature] -= step_size
        this_err = find_err(dataset, d_norm, dp_norm)
        if(this_err < best_err):
            best_err = this_err
            no_improve = 0
            best_rep = deepcopy(dataset.representor)
            continue
        dataset.representor[random_feature] += step_size

        no_improve += 1
        if(no_improve > no_improve_threshold):
            step_size /= 2

    dataset.representor = best_rep
    return best_err


for d_norm, dp_norm in norm_set:
    print(f'for d = {d_norm}, d\' = {dp_norm} error is',
          np.round(find_rep_with_AS(dataset, d_norm, dp_norm), 3))
    print(f'with representor {np.round(dataset.representor, 3)}\n')
```

```
for d = 2.0, d' = 0.0 error is 149.0
with representor [6.2 2.8 4.8 1.8]

for d = inf, d' = 0.0 error is 149.0
with representor [6.4 2.9 4.3 1.3]

for d = 2.0, d' = 1.0 error is 282.34
with representor [5.944 2.914 4.224 1.367]

for d = inf, d' = 1.0 error is 229.426
with representor [5.938 2.812 4.238 1.388]

for d = 1.0, d' = 2.0 error is 42.939
with representor [5.7   3.1   4.    1.191]

for d = inf, d' = 2.0 error is 21.323
with representor [5.758 2.658 3.758 1.175]

for d = 1.0, d' = inf error is 6.05
with representor [5.683 3.05  4.513 1.104]

for d = 2.0, d' = inf error is 3.586
with representor [6.4   2.946 3.7   1.4  ]
```

Second idea

Separately solve for each d', then optimize around that point with alternate search

When $d' = 0$, we want to find a representor that maximizes the errors that are zero, so we should choose the most frequent point

When $d' = 1$, we want to find a representor that minimize the sum absolute of errors, so we should choose the median of points

When $d' = 2$, we want to find a representor that minimize the sum square of the errors, so we should choose the mean of points

When $d' = \infty$, we want to minimize the maximum distance from the representor, so we should choose the middle of our points

In [4]:

```
def find_rep_and_improve(dataset: Dataset, d_norm: int, dp_norm: int):
    step_size, no_improve = 1, 0
    eps, no_improve_threshold = 1e-04, dataset.number_of_feature ** 3

    if(dp_norm == 0):
        dataset.representor = npi.mode(dataset.sample)
    elif(dp_norm == 1):
        dataset.representor = np.median(dataset.sample, axis=0)
    elif(dp_norm == 2):
        dataset.representor = np.mean(dataset.sample, axis=0)
    elif(dp_norm == np.inf):
        dataset.representor = (
            (np.max(dataset.sample, axis=0) - np.min(dataset.sample, axis=0))/2
            + np.min(dataset.sample, axis=0))
    best_err, best_rep = \
        find_err(dataset, d_norm, dp_norm), dataset.representor

    while(step_size > eps):
        random_feature = np.random.randint(0, dataset.number_of_feature)

        dataset.representor[random_feature] += step_size
        this_err = find_err(dataset, d_norm, dp_norm)
        if(this_err < best_err):
            best_err = this_err
            no_improve = 0
            best_rep = deepcopy(dataset.representor)
            continue
        dataset.representor[random_feature] -= step_size

        dataset.representor[random_feature] -= step_size
        this_err = find_err(dataset, d_norm, dp_norm)
        if(this_err < best_err):
            best_err = this_err
            no_improve = 0
            best_rep = deepcopy(dataset.representor)
            continue
        dataset.representor[random_feature] += step_size

        no_improve += 1
        if(no_improve > no_improve_threshold):
            step_size /= 2

    dataset.representor = best_rep
    return best_err


for d_norm, dp_norm in norm_set:
    print(f'for d = {d_norm}, d\' = {dp_norm} error is',
          np.round(find_rep_and_improve(dataset, d_norm, dp_norm), 3))
    print(f'with representor {np.round(dataset.representor, 3)}\n')
```

```
for d = 2.0, d' = 0.0 error is 147.0
with representor [4.9 3.1 1.5 0.1]

for d = inf, d' = 0.0 error is 147.0
with representor [4.9 3.1 1.5 0.1]

for d = 2.0, d' = 1.0 error is 272.292
with representor [5.941 2.908 4.222 1.363]

for d = inf, d' = 1.0 error is 223.216
with representor [5.904 2.804 4.204 1.394]

for d = 1.0, d' = 2.0 error is 42.55
with representor [5.7  3.1  4.   1.18]

for d = inf, d' = 2.0 error is 21.157
with representor [5.791 2.665 3.765 1.207]

for d = 1.0, d' = inf error is 6.05
with representor [6.106 3.075 3.919 1.3  ]
```

```
for d = 2.0, d' = inf error is 3.55
with representor [5.975 2.907 3.95  1.39 ]
```

Thank you very much for taking the time to read this