

Principles of Machine Learning: Exercise 4

Alina Pollehn (3197257), Julian Litz (3362592), Manuel Hinz (3334548)
Felix Göhde (3336445), Felix Lehmann (3177181), Caspar Wiswesser (3221493)
Adrian Köring (3347785), Greta Günther (3326765), Linus Mallwitz (3327653)
Niklas Mueller-Goldingen (3363219), Jennifer Kroppen (2783393)

18.12.2023

Exercise 4.1: Overview

- ① Goal: Minimize $f(\mu) := \frac{1}{n} \sum_{j=1}^n \|x_j - \mu\|$
- ② Alternative formulation: Find $w \geq 0$ s.t. $1^\top w = 1$ and

$$w = \operatorname{argmin}_w \underbrace{\frac{1}{n} \sum_{j=1}^n \|x_j - Xw\|^2}_{=: g(w)}$$

- ③ We want to use the Frank-Wolfe algorithm to solve our optimization problem, therefore we also need the gradient:

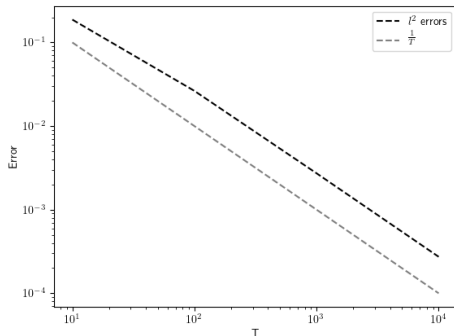
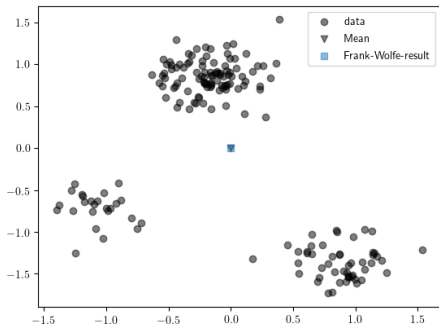
$$\nabla g = 2X^\top X[w - \frac{1}{n}1]$$

Exercise 4.1: Implementation

```
def grad_g(w,n):  
    vecW=np.ones(n)/n  
    return 2*X.T@X@(w-vecW)  
  
def frank_wolfe_minimize_avg_dist(X,T=10000):  
    n=X[0,:].shape[0]  
    wt=np.zeros(n)  
    wt[0]=1  
    for t in range(T):  
        gamma_t=2/(t+2)  
        imin=np.argmin(grad_g(wt,n))  
        wt *= gamma_t  
        wt[imin] += gamma_t  
    return X@wt
```

Exercise 4.1: Error analysis

Because we know the minimizer (which is just the mean of our vectors), we can calculate our error $\epsilon := \|\hat{\mu} - \mu\|$ as a function of the number of iterations. We also know that the error should be in $O(\frac{1}{T})$.



We also ran least squares to estimate the exponent similarly to Task 1-3. This yields $-1.05 \approx -1$, which is also consistent with our theoretical error bound.

Exercise 4.2: Proving two identities

Let X be our data matrix and $w = \frac{1}{n} \mathbf{1}_n$ and $z \in \mathbb{R}^n$.
Then the following identities hold:

$$\begin{aligned} \text{tr}[\overbrace{X^T X}^{A_1} w z^T] &= z^T X^T X w \\ \text{tr}[\underbrace{z w^T X^T X}_{A_2} w z^T] &= z^T z \cdot w^T X^T X w = z^T (z w^T X^T X) w \end{aligned}$$

It suffices to show the following lemma (using either $A = A_1$ or $A = A_2$):

Lemma 1

For a matrix $A \in \mathbb{R}^{n \times n}$, $w = \frac{1}{n} \mathbf{1}_n \in \mathbb{R}^n$ and $z \in \mathbb{R}^n$ the following equality holds:

$$\text{tr}(A w z^T) = z^T A w$$

Proof of Lemma 1

Proof.

- ① First we use $\text{tr}(A \overbrace{wz^T}^{\in \mathbb{R}^{n \times n}}) = \text{tr}(wz^T A)$ (because we apply a cyclic permutation)
- ② Now $a := z^T A$ is just a row vector, therefore

$$\begin{aligned} wa &= \begin{pmatrix} \frac{a_1}{n} & \cdots & \frac{a_n}{n} \\ \vdots & \vdots & \vdots \\ \frac{a_1}{n} & \cdots & \frac{a_n}{n} \end{pmatrix} \in \mathbb{R}^{n \times n} \\ \implies \text{tr}(wa) &= \sum_{i=1}^n \frac{a_i}{n} = \sum_{i=1}^n \frac{a_i}{n} \cdot 1 = \sum_{i=1}^n a_i \frac{(1_n)_i}{n} \\ &= (z^T A)w \end{aligned}$$



k -means clustering: Two perspectives

- ① well known formulation: Minimize:

$$\mu_1, \dots, \mu_k = \operatorname{argmin}_{\mu_1, \dots, \mu_k} \sum_{i=1}^k \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

where $(C_i)_i$ are the clusters and μ_i are the centroids of each cluster.

- ② Alternative formulation:

$$\mu_1, \dots, \mu_k = \operatorname{argmin}_{\mu_1, \dots, \mu_k} \sum_{i=1}^k \sum_{x_j \in C_i} z_{ij} \|x_j - \mu_i\|^2$$

where $z_{ij} \in \{0, 1\}$ and $\sum_{i=1}^k z_{ij} = 1$

- ③ The second formulation can be solved by a Frank-Wolfe-procedure:

```
def FW_KMEANS_VERSION1(X,k,T_max):  
  
    idx=np.random.randint(0,X.shape[1],k)  
    M=X[:,idx]  
    for t in range(T_max):  
        Z = 1/k *np.ones((k,X.shape[1]))  
        Z = FW_UPDATE_Z(X,M,Z, 1)  
        M = X @ Z.T @ np.linalg.inv(Z@Z.T)  
    return M,Z
```

```
def FW_UPDATE_Z(X,M,Z, t_max):  
    ei= np.identity(Z.shape[0])  
    for t in range(t_max):  
        G_z = 2* (M.T @ M @ Z - M.T @ X)  
        #vectorize for loop  
        o=np.argmin(G_z, axis=0)  
        Z += 2/(t+2) * (ei[o].T- Z)  
    return Z
```


Exercise 4.4.1: Results

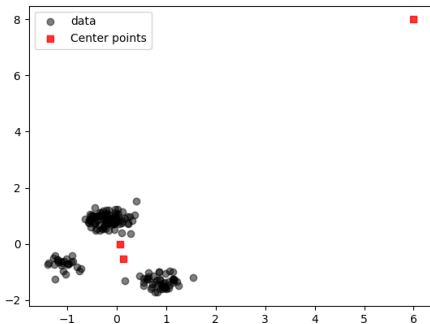


Figure: Result of the naive algorithm

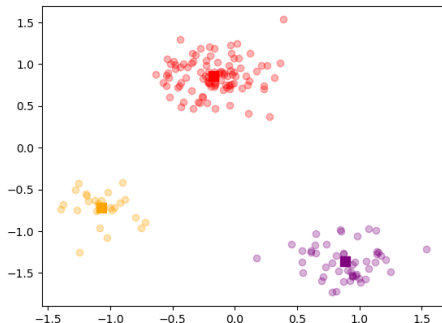


Figure: Result of FW- k -means Version 1

Clearly the results of the naive algorithm are worthless, while also crashing considerably more!

Exercise 4.4.2: Results

- ① While using random data points to construct M causes stability issues, because of two matrix inversions, these were manageable
- ② Given that our solution for Exercise 4.3 is still rather slow for large k , it was still faster to use random, rather than maximally different, data points.



Figure: Result of FW- k -means Version 1

Exercise 4.5:

- ① Code is very similar to exercise 4.4:
 - uses the same vectorization to get rid of the for-loop
 - uses random data points to generate M before the first iteration
- ② Main difference: no matrix inversion \implies more robust
- ③ Similar results: comparable difference when comparing both, or two results of the same version

Exercise 4.5: Results

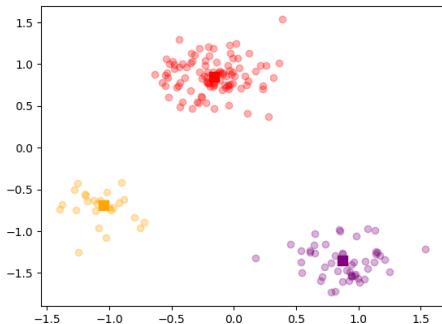


Figure: Result of FW- k -means Version 2 for threeBlobs.csv



Figure: Result of FW- k -means Version 2 for faceMatrix.npy

Exercise 4.6: Comparisson

- 1 Similar optimization problem compared to exercise 4.5: We only drop the constraint

$$Z \in \{0,1\}^{k \times n}$$

- 2 Similar code to exercise 4.5
- 3 Result: We get a set of "exaggerated" representations for each cluster, rather than the mean of each cluster

```
def FW_ARCHETYPAL_ANALYSIS(X,k, T_max=100):  
  
    idx=np.random.randint(0,X.shape[1],k)  
    A=X[:,idx]  
    for t in range(T_max):  
        Z = 1/k *np.ones((k,X.shape[1]))  
        Z = FW_UPDATE_Z(X,A,Z, t_max=100)  
  
        Y = 1/X.shape[1] *np.ones((X.shape[1],k))  
        Y = FW_update_Y(X, Y,Z, t_max=100)  
  
        A = X@Y  
    return A,Y,Z
```

Exercise 4.6: Results

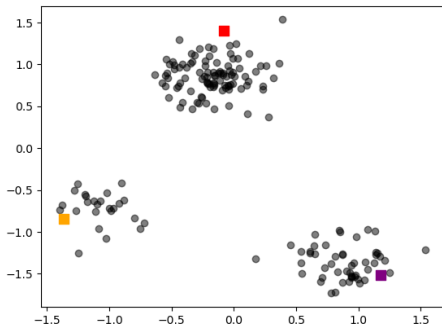


Figure: Archetypes of threeBlobs.csv



Figure: Archetypes of faceMatrix.npy