Assignment 3

Data Mining and Machine Learning (2IIG0)

Sibylle Hess, Ghada Sokar

This is your third assignment of the Data Mining and Machine learning course. The assignment is composed of five questions, the last one is open-ended. Multiple-choice questions have four possible choices; among these, only one answer is correct. The number of points for each question is written at the beginning of it. The maximum achievable score is 100 points. In all the questions, you have to implement the algorithm yourself required in the question using Python unless stated otherwise.

Submit the answers for the first four questions in the **HW3** quiz in Canvas. You have to submit this quiz individually. That is, every group member has to fill out his/her own quiz, even when you are encouraged to work on the exercises together. You can submit the quiz multiple times, the latest one will be graded. Please fill this quiz out carefully.

The open question can be submitted on a group level by means of the **Open Question HW3** assignment in Canvas. That is, one of your group members should upload:

- A PDF document containing the answer to the open question. Include a paragraph about the workload distribution as well. We assume that you can decide yourself how to organize the work in your group. But if someone didn't contribute at all, or only marginally, then you should note this in the PDF.
- A Python file/ notebook containing the implementations you used to answer the implementation-based questions. This notebook is your insurance, it shows what you did and which results you got. If you want to argue for a higher grade/get partial points then you can back it up with the results stated in the uploaded notebook. However, if we see that you couldn't provide a working implementation, but just guessed the correct answer in the MC format, then we will also subtract the corresponding points from your grade.

1. (20 points) **Convolution Neural Network** This exercise is about analyzing the performance of a convolution neural network model that is trained on a part of the handwritten digit dataset (MNIST). The dataset contains 2D grayscale images of digits from 0 to 9. The goal is to learn a function that classifies unseen images to their corresponding targets. In the assignment .zip file, you can find a Python notebook that contains the definition of the architecture, data loading, and the training and test procedures. You are required to analyze the effect of the hyper-parameters on the model performance. To this end:

    1. Explore the functions that are defined in the notebook.
    2. Train the model using the default defined hyper-parameters.
    3. Analyze the performance of the model on the train, validation, and test set.
    4. Train the model with dropout equal to 0.5. Keep the default values for the other hyper-parameters.
    5. Repeat step 3.
    6. Train the model with L2 regularization of 0.05. Keep the default values for the other hyper-parameters.
    7. Repeat step 3.

    Which of the following statements is **correct**?

    A. Using the default hyper-parameters, the model generalizes well since the accuracy on the test set is higher than the accuracy on the validation set.

    ~~B.~~ Using the default hyper-parameters, the model suffers from underfitting.

    C. Using a dropout of 0.5 reduces overfitting.

    ~~D.~~ Using L2 regularization of 0.05 reduces underfitting.

2. (10 points) **PCA Faces** In the accompanying notebook, the Olivetti faces dataset is loaded. The data matrix is here defined such that every row (observation) reflects a picture of a person.

    Implement the PCA algorithm as stated by the pseudocode on the slides (Lecture 9). You can use the sklearn function for computing the truncated SVD or you can compute a full SVD with scipy/numpy and then truncate it manually.

    Use your PCA implementation to decide which of the following statements is **wrong**. We are using the same notation of variables as in the lecture.

    ~~A.~~ The picture represented by $\boldsymbol{\mu}_{\mathrm{F}}$ looks like this:

    B. The reconstructions of the pictures look quite good when we use 100 principal components. Hence, a dimensionality reduction to 100 by PCA is suitable for this dataset.

    The reconstruction of the second picture when using 25, 50 and 100 principal components looks like this (cf. Choice D):

C. The pictures which visualize the first four principal component directions represented by $V_{\cdot\{1,\dots,4\}}$ (indexing starts here at 1) are a selection of the following pictures:

We have two images for every direction (column), because there are two possible vectors which indicate a principal component direction: $V_{\cdot s}$ and $-V_{\cdot s}$. That is, a picture in the first column is the inverted variant of the picture below and vice versa.

D. If we denote with $P = CV_{\cdot\mathcal{R}}$ the low-dimensional $n \times r$-dimensional data representation as returned by PCA, then we can reconstruct image i (cf. Choice B) by

$$P_{i\cdot}V_{\cdot\mathcal{R}}^\top + \boldsymbol{\mu}_\mathbf{F}^\top.$$

3. (10 points) **k-means and SVD Faces** The k-means clustering and the truncated SVD compute a factorization of a given data matrix $D \approx YX^\top$. We use again the Olivetti dataset from the previous exercise as the data matrix.

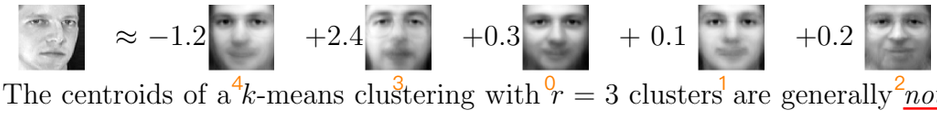
   Implement the functions which return the matrices $X \in \mathbb{R}^{d\times r}$ and $Y \in \mathbb{R}^{n\times r}$, given a data matrix $D \in \mathbb{R}^{n\times d}$ and the rank r, based on the decomposition given by k-means and truncated SVD. For the decomposition of SVD choose $X = V_{\cdot\mathcal{R}}$ as the matrix of right singular vectors. You can use the sklearn k-means function and the sklearn truncated SVD function or you can compute a full SVD with scipy/numpy and then truncate it manually.

   Decide which of the following statements is **wrong**. We are using the same notation of variables as in the lecture.

   A. The rank-r truncated SVD represents every picture in an r-dimensional feature space, where every feature corresponds to one column in the matrix X, which can here be visualized as a face in the original feature space. For example, for $r = 4$ we get the following features:

   B. If the pictures which are represented by the five centroids of k-means look for $r = 5$ like this:

   then the reconstruction of every picture $D_{i\cdot} \approx (YX^\top)_{i\cdot}$ in the dataset is a linear combination of these centroids. For example, the second picture in the dataset is approximated by the following linear combination (rounded to one point after the decimal):



   C. The centroids of a k-means clustering with $r = 3$ clusters are generally *not* a subset of the centroids of a k-means clustering with $r = 5$ clusters.

~~D.~~ The pictures represented by the columns of the matrix X of a rank $r = 3$ truncated SVD are a subset of the pictures represented by the columns of the matrix X of a rank $r = 5$ truncated SVD matrix factorization.

4. (20 points) **k-means Initialization** We will implement now the popular k-means++ initialization technique. Let $dist(i, X) = \min\{\|D_{i\cdot}^\top - X_{\cdot s}\|^2 | 1 \le s \le r\}$ denote the quadratic distance from data point $D_{i\cdot}^\top$ to its closest centroid.

先随机选择一个点作为centroid 相当于此时s=1

先取第i行，然后再转置为列

这里的表示方法与PPT中的相反，但是计算结果是一致的

function INITCENTROIDS(D, r)
  Sample $i \in \{1, \dots, n\}$ uniformly at random
  $X \leftarrow D_{i\cdot}^\top$ X按列存储质心，每个列是一个质心
  $s \leftarrow 2$
  while $s \le r$ **do** 从第二个centroid开始，调用下列算法
    Sample $i \in \{1, \dots, n\}$ with probability $P(i) = \dfrac{dist(i, X)}{\sum_{j=1}^{n} dist(j, X)}$
    $X \leftarrow \begin{bmatrix} X & D_{i\cdot}^\top \end{bmatrix}$ ▷ Attach the new centroid as a column to X
    $s \leftarrow s + 1$
  return X

算法描述：
先随机取一个点作为第一个centroid
循环：
记录每个数据点与其最近的centroid的距离，这个距离越大，则被挑选为下一个centroid的可能性越高。在挑选下一个centroid时，采取带权随机抽样，即可将每个数据被取到的概率当成一个线段，线段长与概率成正比，所有线段构成一长条线。随机在长度范围内取一个点，这个点落入哪个线段，就取哪个点作为centroid

对于这个概率的处理有两种方式，一个是直接取概率最大的，也就是dist最大的点；另一种是取一个[0, 1]随机数，随机数落到哪个概率区间就选哪个点—> 带权重的随机抽样。第二种更好，因为第一种dist最大的点太极端，很容易取到plot中最边界的点

- Implement the initialization of k-means centroids and evaluate the obtained clustering results for multiple initialization runs (you can assess the results visually by means of a plot of the clustering). Around ten runs should be sufficient to get a good impression about the quality of the clustering in average.
- Generate *blobs*, *aniso*, *circles* and *moons* datasets by means of the provided functions in the notebook. Use a parameter setting of $n = 500$ datapoints and an amount of noise `epsilon` $= 0.05$. Choose the number of clusters r as the given ground truth number of clusters.
- Cluster the convex clusters of the blobs and aniso datapoints by k-means, using your implementation of the initialization in the final k-means clustering step.
- Cluster the nonconvex circles and the moons dataset by means of spectral clustering, using your implementation of the initialization in the final k-means clustering step. You can employ the implementation of spectral clustering provided in the Nonconvex Clustering lecture. Use the KNN-neighborhood similarity matrix and try to find for both datasets a suitable number of nearest neighbours $kNN \in \{15, 20, 25, 30\}$.

Which of the following statements is **wrong**?

~~A~~. The k-means++ initialization leads to suitable clusterings of the blobs dataset in most cases.

~~B~~. The k-means++ initialization does **not** lead to suitable clusterings of the *aniso* dataset in most cases.

C. The k-means++ initialization in spectral clustering leads to suitable clusterings of the circles dataset in most cases.

D. The k-means++ initialization in spectral clustering does **not** lead to suitable clusterings of the moons dataset in most cases.

C和D：怎么把k-means的初始和最终centroid点画出来？
如果用老师给的代码就要调用库函数KMeans，这样就不能手动指定initial centroid的坐标

5. (40 points) **Your own Personal Netflix** In the lecture, we have briefly discussed a strategy for recommender systems to cope with the effect of many missing values in the rating matrix. The strategy was to minimize the approximation error *only* for the observed entries. That is, the optimization objective is

$$\min_{X,Y} \|D - O \circ (YX^\top)\|^2 = \sum_{(i,k)\in\mathcal{O}} (D_{ik} - Y_{i\cdot} X_{k\cdot}^\top)^2 \qquad \text{s.t. } X \in \mathbb{R}^{d\times r}, Y \in \mathbb{R}^{n\times r}, \tag{1}$$

(i, k)表示任意一个观测值

where the matrix $O \in \{0, 1\}^{n\times d}$ indicates the observed entries and the set $\mathcal{O} \subseteq \{1, \dots, n\} \times \{1, \dots d\}$ contains the indices of the observed entries in D. Note that the objective above is formulated a bit different than the one stated in the lecture. That is because we assume here that the matrix D has an entry of zero if the corresponding value is missing. In this case, we have $D \circ O = D$ ($\circ$ is the Hadamard multiplication, that is an element-wise multiplication of matrices) and we can write the objective to minimize subject to

lecture中是用mean替代missing value，这里是用的0

Hadamard乘积：矩阵中对应元素相乘

the observed entries as above. However, we have not discussed in detail how to optimize this objective. This will be done in this exercise.

The idea is to apply block coordinate descent on the rows of the factor matrices of X and Y. We can obtain the global minimizers of the objective in Eq. (1) with respect to one row of X or Y (fixing all other coordinates) by means of FONC. First, we compute the partial gradient of the objective in Eq. (1) with respect to coordinate X_{ks}:

$$\begin{aligned}\frac{\partial}{\partial X_{ks}} \sum_{(i,\hat{k})\in\mathcal{O}} (D_{i\hat{k}} - Y_{i\cdot}X_{k\cdot}^\top)^2 &= 2 \sum_{(i,k)\in\mathcal{O}} (D_{ik} - Y_{i\cdot}X_{k\cdot}^\top)(-Y_{is}) \\ &= -2(D_{\cdot k} - YX_{k\cdot}^\top)^\top \operatorname{diag}(O_{\cdot k})Y_{\cdot s}\end{aligned}$$

The diagonal matrix $O_{Xk} = \operatorname{diag}(O_{\cdot k})$ is the diagonal matrix, having the indicator vector of observed entries of feature k on the diagonal. Therewith, we can denote the gradient with respect to one row of X as follows:

$$\begin{aligned}\nabla_{X_{k\cdot}} \sum_{(i,\hat{k})\in\mathcal{O}} (D_{i\hat{k}} - Y_{i\cdot}X_{\hat{k}\cdot}^\top)^2 &= -2(D_{\cdot k} - YX_{k\cdot}^\top)^\top O_{Xk}Y \\ &= -2(D_{\cdot k}^\top - X_{k\cdot}Y^\top)O_{Xk}Y\end{aligned}$$

We compute now the stationary points of this gradient:

$$\begin{aligned}-2(D_{\cdot k}^\top - X_{k\cdot}Y^\top)O_kY &= 0 \\ \Leftrightarrow D_{\cdot k}^\top Y &= X_{k\cdot}Y^\top O_{Xk}Y \\ \Leftrightarrow D_{\cdot k}^\top Y(Y^\top O_{Xk}Y)^{-1} &= X_{k\cdot} \qquad (2)\end{aligned}$$

Now we have here a small problem, since the matrix $Y^\top O_kY$ might not have an inverse. We have already seen how to alleviate this problem in the regression regularization lecture. Likewise, we add here a penalization term to the objective and obtain the penalized objective

$$\min_{X,Y} \|D - O \circ (YX^\top)\|^2 + \lambda\|X\|^2 + \lambda\|Y\|^2 \qquad \text{s.t. } X \in \mathbb{R}^{d\times r}, Y \in \mathbb{R}^{n\times r}. \qquad (3)$$

The penalized objective has now well-defined stationary points

$$D_{\cdot k}^\top Y(Y^\top O_{Xk}Y + \lambda I)^{-1} = X_{k\cdot}. \qquad (4)$$

Likewise, we can compute the minimizers of the objective for a row of Y, resulting in the following block-coordinate descent method for matrix completion.

(b) (5 points) Are the stationary points $X_{k\cdot}$ computed in Eq. (4) actually the minimizers of the block-coordinate objectives

$$\min_{X_{k\cdot}} \|D - O \circ (YX^\top)\|^2 + \lambda\|X\|^2 + \lambda\|Y\|^2 \qquad \text{s.t. } X_{k\cdot}^\top \in \mathbb{R}^r \qquad (5)$$

for $1 \leq k \leq d$? State clearly why or why not this is the case.

```
function MatrixCompletion(D, r; t_max = 100, λ = 0.0001)
    (X, Y) ← InitRandom(n, d, r)
    O ← IndicatorNonzero(D)
    t ← 1
    while t < t_max do
        for k ∈ {1, ..., d} do
            O_Xk ← diag(O_1k, ..., O_nk)
            X_k· ← D_·k^⊤ Y (Y^⊤ O_Xk Y + λI)^{-1}
        for i ∈ {1, ..., n} do
            O_Yi ← diag(O_i1, ..., O_id)
            Y_i· ← D_i· X (X^⊤ O_Yi X + λI)^{-1}
        t ← t + 1
    return (X, Y)
```

In the zip-file for this exercise is a small movie-lens dataset. You will find predefined implementations in the accompanying notebook to create a data matrix D, reflecting a small subset of the users and movies. Unless stated otherwise, we use as default parameter setting $t_{max} = 100, r = 5, \lambda = 0.00001$.

(a) (20 points) Implement the function MatrixCompletion as outlined in the Pseudocode above. Plot the iterations t against the Mean Squared Error on the Observed (MSEO) entries for the corresponding iterate (X, Y) on the MovieLens data:

$$MSEO(X, Y) = \frac{1}{|O|}\|D - O \circ (YX^\top)\|^2.$$

(b) (5 points) Are the stationary points $X_{k\cdot}$ computed in Eq. (4) actually the minimizers of the block-coordinate objectives

$$\min_{X_{k\cdot}} \|D - O \circ (YX^\top)\|^2 + \lambda\|X\|^2 + \lambda\|Y\|^2 \qquad \text{s.t. } X_{k\cdot}^\top \in \mathbb{R}^r \tag{5}$$

for $1 \leq k \leq d$? State clearly why or why not this is the case.

Yes, because the objective is still convex

(c) (5 points) Choose a suitable stopping criterion which checks for (approximate) convergence of the iterates. How many iterations does it take until convergence is reached according to your stopping criterion and what is the resulting MSEO in comparison to the MSEO obtained for 100 iterations?

(d) (10 points) State for $\lambda \in \{1, 0.5, 0.1, 0.0001\}$ the recommendations for the first three users. Comment on the effect which the regularizing parameter has on the result in terms of interpretability and the obtained MSEO. Which value would you choose for λ and why?