Assignment 3

Data Mining and Machine Learning (2IIG0)

Stiven Dias, Sibylle Hess and 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 28×28 grayscale images of digits from 0 to 9. The goal is to learn a function $h : \{0, 1, \ldots, 255\}^{784} \mapsto \{0, 1, \ldots, 9\}$ that classifies unseen images to their corresponding targets. In the assignment files, you can find a Python notebook (HW3.ipynb) that contains the definition of the architecture, data loading, and the training and test procedures. Please use the last stable versions of pytorch (1.13.1) and torchvision (0.14.1) to ensure the correct results are reproduced. 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. (15 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. 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.

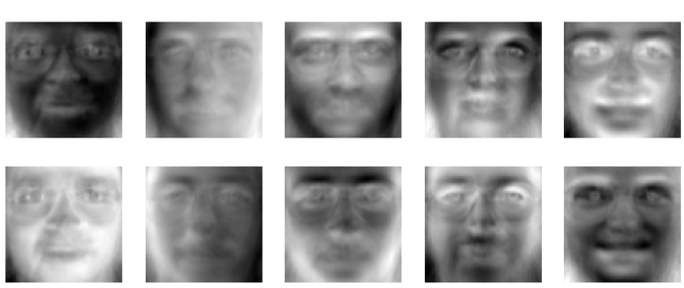
   As discussed in the lecture, PCA is based on a low-rank matrix factorization, where

   $$D - \mathbf{1}\mu_{\mathtt{F}}^{\top} = C = YX^{\top},$$

   where X indicates the directions of maximum variance, i.e., the PCs. Y indicates the low-dimensional representations of the data points. Note, that when we want to reconstruct the data matrix based on the low-rank approximation, we have to add the mean $\mu_{\mathtt{F}}$ to the matrix product.

   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 pictures which visualize the first four principal component directions represented by $X_{\cdot\{1,\ldots,4\}}$ (mathematical indexing starts at 1, that is in Python the range will go from zero to four) 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: $X_{\cdot s}$ and $-X_{\cdot s}$. That is, a picture in the first column is the inverted variant of the picture below and vice versa.

B. The first picture (index `0` in Python) has a low-dimensional representation of $Y_{1\cdot} = (-0.074, -0.011, -0.029, 0.032, 0.076)$ (rounded to three decimals after the point).

C. The reconstruction of the picture number 20 (Python index `19`) when using 25, 50 and 100 principal components looks like this:



D. We can also generate pictures: the face with the low-dimensional representation given by the negative constant one vector $(-1, -1, \ldots, -1) \in \{-1\}^{50}$ (using 50 principal components) looks like this:

3. (25 points) **k-means Initialization** Implement the popular greedy k-means++ initialization technique. Given a matrix $X \in \mathbb{R}^{d \times r}$, let $dist(v, X) = \min\{\|v - X_{\cdot s}\|^2 | 1 \leq s \leq r\}$ denote the quadratic distance from vector v to the closest column vector in X.

function initGreedyKMeans(D, r, l)
  Sample $i_1, \ldots, i_l \in \{1, \ldots, n\}$ uniformly independently at random
  $i \leftarrow \arg\min_{i \in \{i_1, \ldots, i_l\}} \sum_{j=1}^{n} \|D_{j\cdot} - D_{i\cdot}\|^2$
  $X \leftarrow D_{i\cdot}^\top$
  $s \leftarrow 2$
  while $s \leq r$ **do**
    Sample $i_1, \ldots, i_l \in \{1, \ldots, n\}$ independently with probability P, where
    $P(i) = \dfrac{dist(D_{i\cdot}^\top, X)}{\sum_{j=1}^{n} dist(D_{j\cdot}^\top, X)}$ for $i \in \{1, \ldots, n\}$
    $i \leftarrow \arg\min_{i \in \{i_1, \ldots, i_l\}} \sum_{j=1}^{n} dist(D_{j\cdot}^\top, \begin{bmatrix} X & D_{i\cdot}^\top \end{bmatrix})$
    $X \leftarrow \begin{bmatrix} X & D_{i\cdot}^\top \end{bmatrix}$ ▷ Attach the new centroid as a column to X
    $s \leftarrow s + 1$
  return X

- Implement the greedy k-means++ initialization. Make sure to use the predefined function framework from the accompanying notebook which sets the seed of the random generator. Make sure that your numpy and sklearn versions are updated. You have to use the versions denoted in the notebook or the random generator might give different results.
- Generate the *moons* datasets as provided in the notebook. Use a parameter setting of $n = 500$ datapoints and an amount of noise `epsilon` $= 0.05$.
- Cluster the moons dataset by means of spectral clustering, using your implementation of the initialization for the k-means clustering step. Use the implementation of spectral clustering provided in the notebook. Use the KNN-neighborhood similarity matrix with various numbers of nearest neighbours $kNN \in \{15, 25, 30, 35\}$. Set the k-means (initialization) parameters to $r = 2$ (the number of ground truth clusters) and $l = 10$.

- Evaluate the obtained clustering results by means of the Normalized Mutual Information (NMI) score, as provided by sklearn.

(a) (5 points) Interpretation of the greedy k-means++ technique. Which of the following statements is **wrong**?

A. The first centroid in line 3 is chosen as the candidate that minimizes the 1-means problem ($r = 1$) the most.

B. The probability that data point $D_{i\cdot}$ is sampled as a centroid candidate in line 7 is higher, the further away the data point is from all centroids chosen so far.

C. For a given centroid matrix $X \in \mathbb{R}^{d\times r}$, we have the following equality:

$$\sum_{i=1}^{n} dist(D_{i\cdot}^\top, X) = \min_Y \|D - YX^\top\|^2 \quad \text{s.t. } Y \in \mathbb{1}^{n\times r}$$

D. The selection of the centroid in line 9 is equivalent to choosing the centroid among the candidates with maximum probability: $i \leftarrow \arg\max_{i\in\{i_1,\dots,i_l\}} P(i)$

(b) (20 points) State for each value of $kNN \in \{15, 25, 30, 35\}$ the NMI score that you obtain for the computed clustering. Round the NMI to two decimals after the point.

4. (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

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

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 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:

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

Now we have here a small problem, since the matrix $Y^\top O_k Y$ 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

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

The penalized objective has now well-defined stationary points

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

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.

```
1: function MatrixCompletion(D, r; t_max = 100, λ = 0.0001)
2:     (X, Y) ←InitRandom(n, d, r)
3:     O ←IndicatorNonzero(D)
4:     t ← 1
5:     while t < t_max do
6:         for k ∈ {1, ..., d} do
7:             O_Xk ← diag(O_1k, ..., O_nk)
8:             X_k· ← D_·k^⊤ Y (Y^⊤ O_Xk Y + λI)^{-1}
9:         for i ∈ {1, ..., n} do
10:            O_Yi ← diag(O_i1, ..., O_id)
11:            Y_i· ← D_i· X (X^⊤ O_Yi X + λI)^{-1}
12:        t ← t + 1
13:    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 $1 \le k \le d$? State clearly why or why not this is the case.

(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?