

NOTE: Problems involving their own coding will be uploaded as a separate file. The outputs will be placed here

PCA

Problem 1. [30 points] In this assignment, you will explore PCA as a technique for discerning whether low-dimensional structure exists in a set of data and for finding good representations of the data in that subspace. To this end, you will do PCA on a Iris dataset which can be loaded in `scikit-learn` using following commands:

```
from sklearn.datasets import load_iris iris = load_iris()
X = iris.data
y = iris.target
```

1. Carry out a principal component analysis on the entire **raw** dataset (4-dimensional instances) for $k = 1, 2, 3, 4$ components. How much of variance in data is retained when $k = 1, 2, 3, 4$, respectively?
2. Apply the centering and standardization operations from Problem 1 on raw data set and repeat part (1) on processed data. Explain any differences you observe compared to part (1) and justify.
3. Project the raw four dimensional data down to a two dimensional subspace generated by first two top principle components (PCs) from part (2) and produce a scatter plot of the data. Make sure you plot each of the three classes differently (using color or different markers). Can you see the three Iris flower clusters?
4. Either use your k-means++ implementation from previous homework or from `scikit-learn` to cluster data from part (3) into three clusters. Explain your observations.

1

Results:

```
n_components = 1

Explained Variance Ratio: [0.92461872]
Explained Variance: [4.22824171]

n_components = 2

Explained Variance Ratio: [0.92461872 0.05306648]
Explained Variance: [4.22824171 0.24267075]

n_components = 3

Explained Variance Ratio: [0.92461872 0.05306648 0.01710261]
Explained Variance: [4.22824171 0.24267075 0.0782095 ]

n_components = 4

Explained Variance Ratio: [0.92461872 0.05306648 0.01710261 0.00521218]
Explained Variance: [4.22824171 0.24267075 0.0782095 0.02383509]
End of part 1
```

```
Processed Data, n_components = 1

Explained Variance Ratio: [0.928563]
Explained Variance: [0.19419117]

Processed Data, n_components = 2

Explained Variance Ratio: [0.928563  0.04816961]
Explained Variance: [0.19419117 0.01007375]

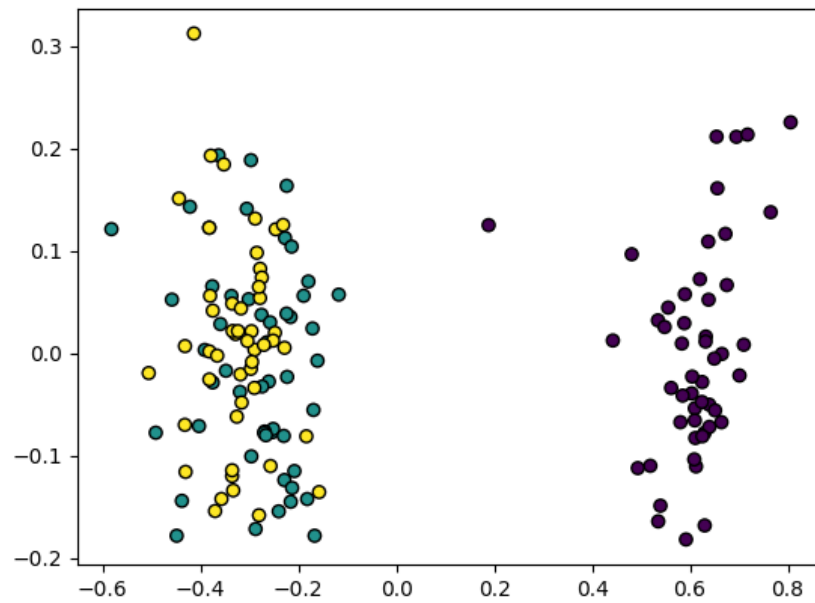
Processed Data, n_components = 3

Explained Variance Ratio: [0.928563  0.04816961 0.01516153]
Explained Variance: [0.19419117 0.01007375 0.00317074]

Processed Data, n_components = 4

Explained Variance Ratio: [0.928563  0.04816961 0.01516153 0.00810585]
Explained Variance: [0.19419117 0.01007375 0.00317074 0.00169518]
End of part 2
```

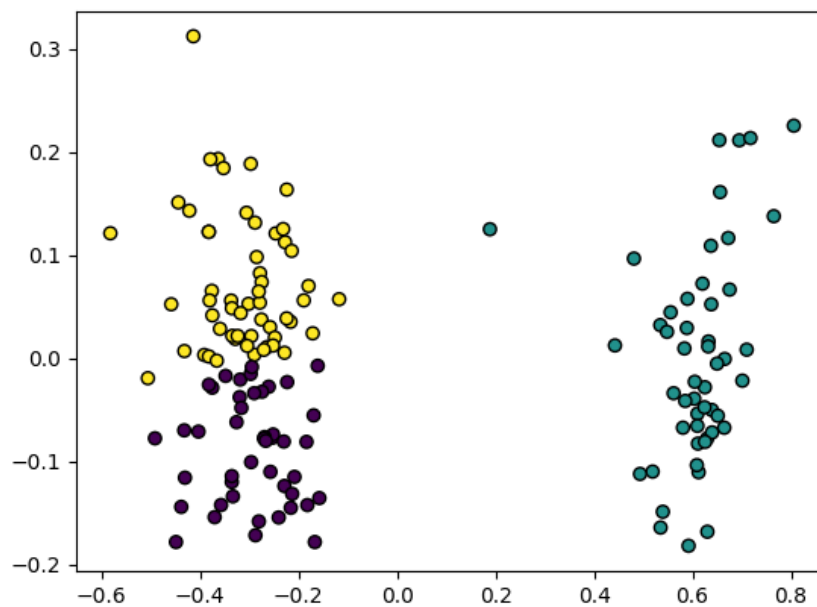
```
Explained Variance Ratio: [0.928563  0.04816961]
Explained Variance: [0.19419117 0.01007375]
c:\CMPSC 448\HW5\Homework5\test\lib\site-packages\sklearn\c
warning
warnings.warn(
End of part 3
```



Part 3 graph:



$x=-0.362$ $y=0.085$



Part 4 graph:



$x=-0.372$ $y=0.249$

Matrix Factorization

Problem 2. [30 points] Recall the following objective for extracting latent features from a partially observed rating matrix via matrix factorization (MF) for making recommendations, discussed in the class:

$$\min_{\mathbf{U}, \mathbf{V}} \left[f(\mathbf{U}, \mathbf{V}) \equiv \sum_{(i,j) \in \Omega} (r_{i,j} - \mathbf{u}_i^\top \mathbf{v}_j)^2 + \alpha \sum_{i=1}^n \|\mathbf{u}_i\|_2^2 + \beta \sum_{j=1}^m \|\mathbf{v}_j\|_2^2 \right] \quad (1)$$

where

- n : number of users
- m : number of items
- $r_{i,j}$: (i, j) -th element in $\mathbf{R} \in \mathbb{R}^{n \times m}$ input partially observed rating matrix
- $\Omega \subseteq [n] \times [m]$: index of observed entries in rating matrix, where $[n]$ denotes the sequence of numbers $\{1, 2, \dots, n\}$.
- k : number of latent features
- $\mathbf{U} \in \mathbb{R}^{n \times k}$: the (unknown) matrix of latent feature vectors for n users (the i th row $\mathbf{u}_i \in \mathbb{R}^k$ is the latent features for i th user)
- $\mathbf{V} \in \mathbb{R}^{m \times k}$: the (unknown) matrix of latent feature vectors for m items (the j th row $\mathbf{v}_j \in \mathbb{R}^k$ is the latent features for j th movie)



Please do the followings:

1. In solving Equation 1 with iterative Alternating Minimization algorithm (fixing $\mathbf{V}^{(t)}$ and taking gradient step for $\mathbf{U}^{(t)}$ and vice versa), discuss what happens if $\mathbf{U}^{(0)}$ and $\mathbf{V}^{(0)}$ are initialized to zero?
 2. Discuss why when there is no regularization in basic MF formulated in Equation 1, i.e., $\alpha = \beta = 0$, each user must have rated at least k movies, and each movie must have been rated by at least k users (Hint: consider the closed form solution for \mathbf{u}_i and \mathbf{v}_j in notes and argue why these conditions are necessary without regularization).
 3. Computing the closed form solution in part (2) could be computational burden for large number of users or movies. A remedy for this would be using iterative optimization algorithms such as Stochastic Gradient Descent (SGD) where you sample movies in updating the latent features for users and sample users in updating the latent features of movies. Derive the updating rules for $\mathbf{u}_i^{(t)}$ and $\mathbf{v}_j^{(t)}$ at t th iteration of SGD. Show the detailed steps.
-
1. Then there would be no weight to any of the vectors. Without the weight, we are unable to calculate the rating of the matrix
 2. Regularization in machine learning involves the process of minimizing the adjusted loss function and prevent overfitting or underfitting. In the above equation, regularization exists in some capacity. By taking users' data for what movie they like and using \mathbf{U} and \mathbf{V} to calculate the rating

How you rate a movie?

Let assume we have k factors in total for all users

We can represent the user's and movie's weights for all factors as a vector:

$$\mathbf{u} = \begin{bmatrix} u_1(\text{weight for 1st factor}) \\ u_2(\text{weight for 2nd factor}) \\ \vdots \\ u_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k \quad \mathbf{v} = \begin{bmatrix} v_1(\text{weight for 1st factor}) \\ v_2(\text{weight for 2nd factor}) \\ \vdots \\ v_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k$$

Then, the rating of the user for the movie is just:

$$\text{RATING}_{\text{user, movie}} = \mathbf{u}^\top \mathbf{v} = \sum_{f=1}^k u_f v_f$$

25

as shown here:

Basically, U and V are the factors that create regularization for the above function because the weight of the vectors determine the how the models will look like, regularization doesn't exist as we usually know it in this function because it already exists in some capacity and with the weight of these vectors we no longer need to worry about overfitting or underfitting because we can simply calculate what the model will look like with the rating function since each movie must be rated by K users

$$3. \text{ Gradient for } u_i: \frac{df}{du_i} = \sum_{j \in [m], i, j \in \Omega} (R_{i,j} - \mu_i^T v_j)(-v_j) + 2\alpha u_i$$

Update rule for $u_i^{(t)}$ at t^{th} iteration of SQD:

$$u_i^{(t)} = u_i - n_t (2(u_i^t v_j - R_{ij})(v_j) + 2\alpha u_i)$$

$$\text{Gradient for } v_j: \frac{df}{dv_j} = \sum_{i \in [m], i, j \in \Omega} (R_{i,j} - \mu_i^T v_j)(-u_i) + 2\beta v_j$$

Update rule for $v_j^{(t)}$ at t^{th} iteration of SQD:

$$v_j^{(t)} = v_j - n_t (2(u_i^t v_j - R_{ij})u_i + 2\beta v_j) \text{ Sorry } N = \eta \text{ my apologies}$$

MDP and RL

Problem 3. [40 points] Consider the MDP in Figure 1. There are five states $\{A, B, C, D, T\}$ and two actions $\{a_1, a_2\}$. The state T is the terminal state. The numbers on each arrow show the probability of moving to the next state and the reward, respectively. For example, if the agent takes action a_1 at state A , with probability 0.7 it will move to state B and will be rewarded -10, and with probability 0.3 it will move to state C and will be rewarded -5.

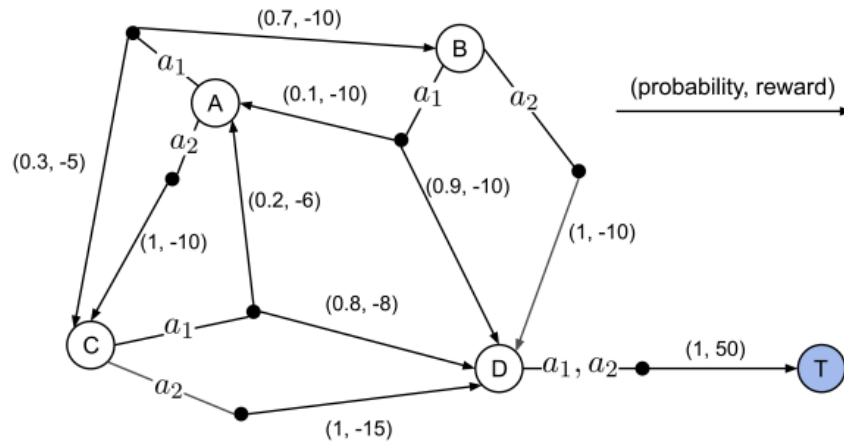


Figure 1: MDP

(1) [10 points] For a policy π that always takes action a_1 at every state, write down the Bellman Equation for state value function for each state using discount factor γ .

(2) [10 points] From the Bellman Equation in (1), compute the state values using $\gamma = 0.5$.

(3) [5 points] Consider a random policy π_0 that uniformly selects actions at each state (i.e., the probability of taking each of two actions under this policy is 0.5). Initialize $v_0(s) = 0$ for all states, apply one sweep of policy evaluation to show $v_1(s)$ for all states using $\gamma = 1$.

(4) [5 points] Based on the result of (3), apply policy improvement by greedification with $\gamma = 1$ to show the improved policy π_1 . Select actions uniformly if there is a tie.



Figure 2: An Episode for MDP in Figure 1.

Now let's do Temporal Difference Control on this MDP using SARSA and Q-Learning. We initialize $Q(s, a) = 0$ for all pairs of (s, a) , and we use step size $\alpha = 0.1$ and discount factor $\gamma = 0.5$. Suppose we sample an episode in Figure 2.

(5) [5 points] Use SARSA to update Q values by processing the episode from beginning to end.

(6) [5 points] Use Q-Learning to update Q values by processing the episode from beginning to end.

1. 1st Policy π_1 2nd Policy π_2

$s_0 \rightarrow a_1$

$s_0 \rightarrow a_1$

$s_1 \rightarrow a_2$

$s_1 \rightarrow a_1$

$s_2 \rightarrow a_2$

$s_2 \rightarrow a_1$

$s_3 \rightarrow a_1$

$s_3 \rightarrow a_2$

$$V_{\pi}(s) = \sum_a^{\pi}(a|s) \sum_{s',r}^p(s',r|s,a)[r + \gamma v_{\pi}(s')]$$

2. $V_{\pi}(s_0) = 0.7 \pm 10(0.5) = -3.5$, $V_{\pi}(s_1) = 0.1 \pm 10(0.5) = -4.9$, $V_{\pi}(s_2) = 1 \pm 10(0.5) = -4$, $V_{\pi}(s_3) = 1 \pm 15(0.5) = -6.5$ NOTE: \pm means adding a negative number ie $0.7 + -10(0.5)$ or $0.7 - 10(0.5)$ apologies for the typo

3. $V_0(s) = 0 \Rightarrow V_1(s) = 0.5(0.7 \cdot [-10 + 0] + 0.1 \cdot [-10 + 0]) + 0.5(1 \cdot [-10 + 0] + 1 \cdot [-15 + 0])$
4. $V_\pi(s_0) = 0.7 - 10(1) = -9.3$ while the previous answer for policy 0 was -3.5 . Therefore the policy as improved
1. **Q-Learning:** $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
2. **SARSA:** $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
5. $0 + 0.1(-6 + 0.5(-10 - 0)) = -1.1$
6. $0 + 0.1(-6 + 0.5(50 - 0)) = 1.9$

Bonus

Problem 4. [20 points] As we discussed in the lecture, the PCA algorithm is sensitive to scaling and pre-processing of data. In this problem, we explore few data pre-processing operations on data matrices.

1. Explain what does each of the following data processing operation mean, why it is important, and how you apply to a data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, n samples each with d features ¹ (if it helps, use mathematical equations to show the calculations).
 - (a) Centering or mean removal
 - (b) Scaling of a feature to a range $[a, b]$
 - (c) Standardization
 - (d) Normalization (between 0 and 1)
2. Apply the above operations to the following data matrix with 5 samples and two features independently and show the processed data matrix. You will have 4 different matrices based on each processing operation. For scaling pick $a = 0$ and $b = 1$.

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ -1 & 1 \\ 0 & 1 \\ 2 & 4 \\ 3 & 1 \end{bmatrix}$$

1.
 - a. Centering or mean removal is an operation that simply center data by removing the average value of each characteristic, then it scales the results by dividing non-constant characteristics by their standard deviation. It is usually beneficial to remove the mean from each feature so that it's center:

High-dimensional feature mapping in kernel methods

Centering training data $x_i \rightarrow (x_i - \mu)$, where $\mu = \frac{1}{n} \sum_{i=1}^n x_i$

$$x_{scaled} = \frac{x - \hat{mean}}{sd}$$

- b. Scaling of a feature to a range [a, b] also known as min-max scaling is a technique used in machine learning to standardize independent features present in data in a fixed range. It involves running the data via pre-processing to handle highly varying magnitudes depending on the values or units given:

Also known as min-max scaling or min-max normalization, rescaling is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [-1, 1]. Selecting the target range depends on the nature of the data. The general formula for a min-max of [0, 1] is given as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- c. Standardization: The process of rescaling one or more attributes so that the data presented has a mean of 0 and a standard deviation of 1: $x_{new} = (x - \text{mean}) / \text{std}$
- d. Normalization (between 0 and 1): Technique often applied as a part of data preparation for machine learning. The goal is to change the values of numeric columns in the dataset to use a common scale without changing the difference in the ranges of values preventing loss of information
2. The above matrix was applied to an algorithm written in python containing said operations, the file will be uploaded separately. These are the results produced, each one corresponds to each operation respectively:

```
"C:\CMPSC 448\HW5\Homework5\test\Scripts\python.exe" "C
Mean standardized data: A    1.0
B    1.8
dtype: float64
Standard Deviation standardized data: A    1.581139
B    1.303840
dtype: float64
Mean standardized data: [0.0000000e+00  4.4408921e-17]
Standard Deviation standardized data: [1.  1.]

Process finished with exit code 0
```