ECE 479 - Homework 1

Ahmadreza Eslaminia

Ae15

Question 1: Edge Computing and Sensors

1.

Cloud Computing:

Advantage:

a) It's cost saving in comparison to in-house data center and edge devices.
b) It can be accessed globally just through an internet connection.

Disadvantage:

a) The distance between the cloud and devices can cause high latency.
b) Storing sensitive data in the cloud can pose security risks, especially if the data is not properly secured.

Edge Computing:

Advantage:

a) It provides lower latency by decreasing the distance that the data has to travel.
b) Edge computing can continue to operate even if the connection to the cloud is lost.

Disadvantage:

a) Edge computing systems can be more expensive to set up and maintain than cloud-based systems.
b) Edge computing systems have limited resources compared to the cloud, which can limit their ability to handle large amounts of data.

2. Cloud computing or edge computing? Why?
   a) Health data collected by a smartwatch to track your daily activities: In this application, edge computing is more suitable since it would provide lower latency. In addition, health data can be considered as sensitive data that is needed to be secure, so edge computing will provide more security.
   b) Temperature sensors are placed inside a refrigerated storage container to regulate the temperature during the shipping process: In this application it is a trade-off between how much scalability we want and how much internet access we have in the shipping process.

If scalability and global access and monitoring is important for us, we should use a cloud-based computing platform. However, if the internet connection is not reliable enough in the shipping routes it is not reasonable to use cloud computing.

c) License plate readers at toll plazas: For this application, it would be better to use cloud computing because the scalability of the system is important for us. We need all these systems get connected and for that purpose using internet is reasonable. In addition, using cloud is more cost effective rather than using lots of edge devices.

d) Medical wearable devices that detect when you fall: In this application, it is better to use edge computing since the latency is so important in this case. Also, edge computing provides more reliability than cloud computing.

3. WSN aggregation strategy:

First we will find the Dijkstra's algorithm for node K,E, and G to find their best ways to the A.

$Dijkstra$ (K, $SinkNode$) = {K, I, B, A}

$Dijkstra$ (G, $SinkNode$) = {G, F, E, D, B, A}

$Dijkstra$ (E, $SinkNode$) = {E, D, B, A }

So as we can see we should aggregate the E & G node's data in the E node since it is the earliest node in the pass. Then we can aggregate E&G node data with the K's node data at the B node.

Also, as we can see the aggregation time is lower than the minimum of the latencies between each two different nodes, so it is reasonable to aggregate data as much and as soon as possible.

# of Operation	K	E	G	B	
Transmission	2	2	2	1	
Aggregation	0	1	0	1	
Total	2	3	2	2	

Time (ms)	K	E	G	B	whole
Transmission	55	20	25	20	120
Aggregation	0	5	0	5	10
Total	55	25	25	25	130

Question 2:

1. **Python one-liners:**

a) Sum (h**2 for h in range(i))
b) sum (A[3*i] for i in range ((len(A) + 2)//3))
c) sum (A[3*i+2] for i in range ((len(A))//3))
d) B = A[::-1]
e) C = list(zip(A, B))
f) C = [x for x in A if x in B]

2. <u>More one-liners: List and String Operations:</u>

a) s0_list = s0.split()
b) count = s0.count('coffee')
c) s0_list = s0.split()[0:4]
d) s0_list.remove('some')
e) s0_list.insert(s0_list.index('drinks')+1, 'much')
f) s0_list.append('everyday')
g) s1_list = s1.split()
h) s0_list.extend(s1_list)
i) F = " ".join(s0_list)

3. **<u>NumPy Slicing</u>**

tmp shape is (2, 2, 2, 1).
For i we have row 1 & row 2
For j we have row 0,1
For k we have row 0,1
For l we have row 0

4. **<u>NumPy Operations</u>**

B can be found by dividing the elements of the output by the corresponding elements of A.
B = output / A
B = [2. 7. 3. 5. 6. 4.]

```
import numpy as np
A = np.array([1, 2, 3, 4, 5, 6])
B = np.array([2, 7, 3, 5, 6, 4])

A1 = np.reshape(A, [2,3])
B1 = np.reshape(B, [3,2])
print(np.dot(A1.T, B1.T).ravel())
```

Question 3: ML Concepts

1. **Supervised vs. Unsupervised Learning**
   a) In the supervised learning our goal is to seek a hard-to-find feature using easy-to-find ones with specific labels. In contrast in the unsupervised learning, we cannot exactly define what we are looking for and there is no label.
   b) Supervised or Unsupervised??
      a. Identify whether a person belongs to this company, using facial recognition: In this case we are going to use Supervised learning since the label for whether a person is in the company or not is known already.
      b. Find how many different customer groups are visiting a shopping: In this problem we do not have exact group types (labels) so we have to use some unsupervised methods(clustering) for this purpose
      c. Approximate the daily schedule of a certain person: In this case, the labeled dataset would consist of data about the person's daily activities and the corresponding time-stamps, and the goal is to use this data to predict the person's future schedule. So, it is going to be a supervised problem.
      d. Predict the market price of a house in the neighborhood, using historical data: Since we have access to the previous data with knowledge of our goal, which is the price this is going to be a supervised problem.
      e. Enhance the details of a blurry image with machine learning: This problem can be solved through supervised learning, the model is trained on a labeled dataset where the desired output is already known, such as pairs of blurry and sharp images. The goal is to use the sharp images as ground truth to restore the details in the blurry images.

c) KNN (K-Nearest Neighbor) is a supervised learning algorithm. In this classifier, the model is trained on a labeled dataset where the desired output is already known.
K-means is an unsupervised learning algorithm. The K-means algorithm groups similar data points together into K clusters, where K is a user-defined parameter. So, data is not labeled, and we have to discover hidden patterns or relationships within the data. So, it is a unsupervised method.

2. Fitting the data:
   a) Among these models, model #2 (green one) is better than other models since it is not too simple such first model to miss the important information of the data pattern (underfitting) and it is not too complex such as model three that is too specific to the current data pattern and is going to have high test error though the training error is low(overfitting)
   b) The first model is going to have high test and training error since the model is too simple. It is called underfitting of the model. The third model is too complex and is too fitted to the training data. This would cause that model cannot perform well on the new data since it is specific to current data. This issue is called overfitting.
3. Training and Validation:
   a) We have to divide the training and testing data to prevent overfitting of the model accuracy that we report. If the model sees the test data before training the model so we cannot measure its performance when it sees new data. For example, consider in KNN if we do not partition data into test and train datasets the test accuracy would be 100 percent.
   b) Testing set is used to evaluate the real-world performance of a model, but it has limitations. One limitation is that the performance of a model may be dependent on the particular split of the data into training and testing sets. This can lead to unstable results and a higher variance in the evaluation of the model. To mitigate this, validation or cross-validation is used. Cross-validation helps to ensure that the model generalizes well to new

data and to reduce the risk of overfitting to the testing set. By evaluating the model on multiple validation sets, we can get a more reliable estimate of its real-world performance. Tuning the model based on testing accuracy is not recommended because it can lead to overfitting to the testing set.

Question 4: PCA

1. The curse of dimensionality refers to the challenge of modeling high-dimensional data, where the number of features (dimensions) is much greater than the number of samples. In high-dimensional data, the number of dimensions can be so large that it becomes difficult to find meaningful patterns and relationships between the features. One reason is because the data would become too sparse.

2. The statistical interpretation of PCA involves finding the eigenvectors and eigenvalues of the covariance matrix of the data, which represents the relationships between the variables. These eigenvectors, known as principal components, are used to project the data into a lower-dimensional space, where each principal component captures a portion of the variance in the data.

   The first principal component is chosen to be the direction in which the data varies the most, and the subsequent components are chosen to be orthogonal to the previous ones and capture the remaining variance in the data. The proportion of variance captured by each component can be represented by the corresponding eigenvalue, which allows us to rank the importance of each component in explaining the variation in the data.

3. Kir

4. PCA with NumPy:

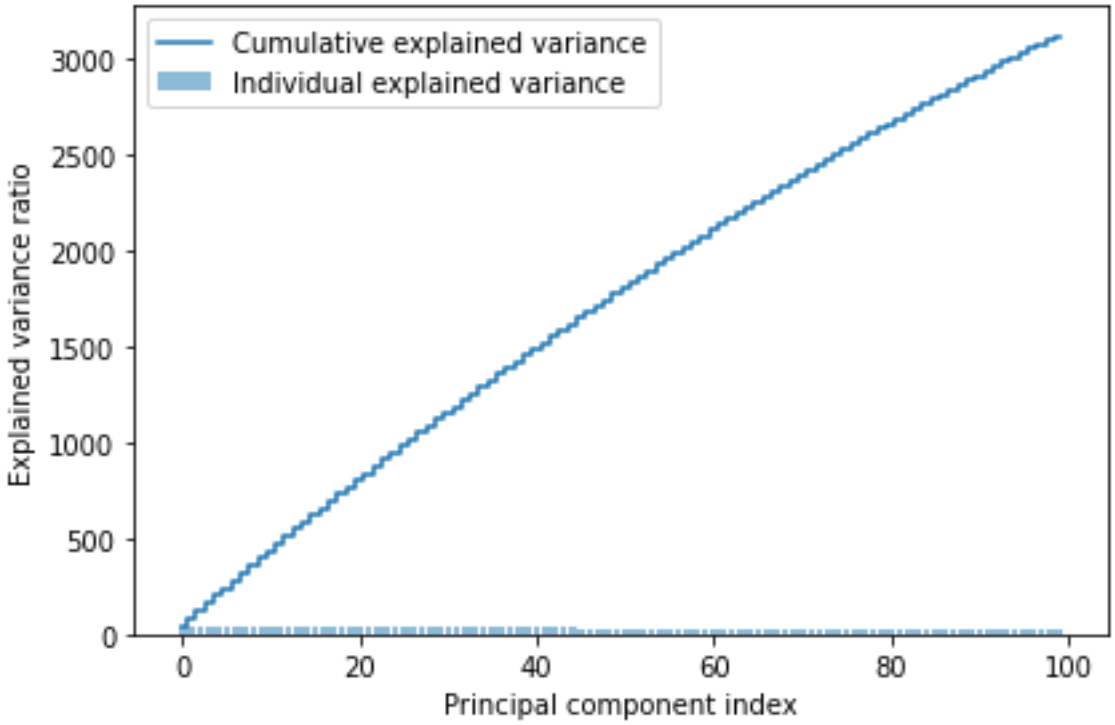
a) data = np.random.randn(1000 ,100)
mean = np.mean(data, axis=0)
std = np.std(data, axis=0)
data = (data - mean) / std

b) U, S, Vt = np.linalg.svd(data)
U is the left singular matrix, which represents the transformed data in the new coordinate system defined by the principal components.
S is a 1-dimensional array of singular values, which are the square roots of the eigenvalues of the covariance matrix of the data. The singular values represent the magnitude of the variance explained by each principal component.
Vt is the transposed right singular matrix, which represents the principal components as the eigenvectors of the covariance matrix. Each row of Vt corresponds to a principal component, and the columns represent the weights of each feature in the calculation of the principal component.
In essence, U represents the new transformed data, S represents the explained variance, and Vt represents the principal components.

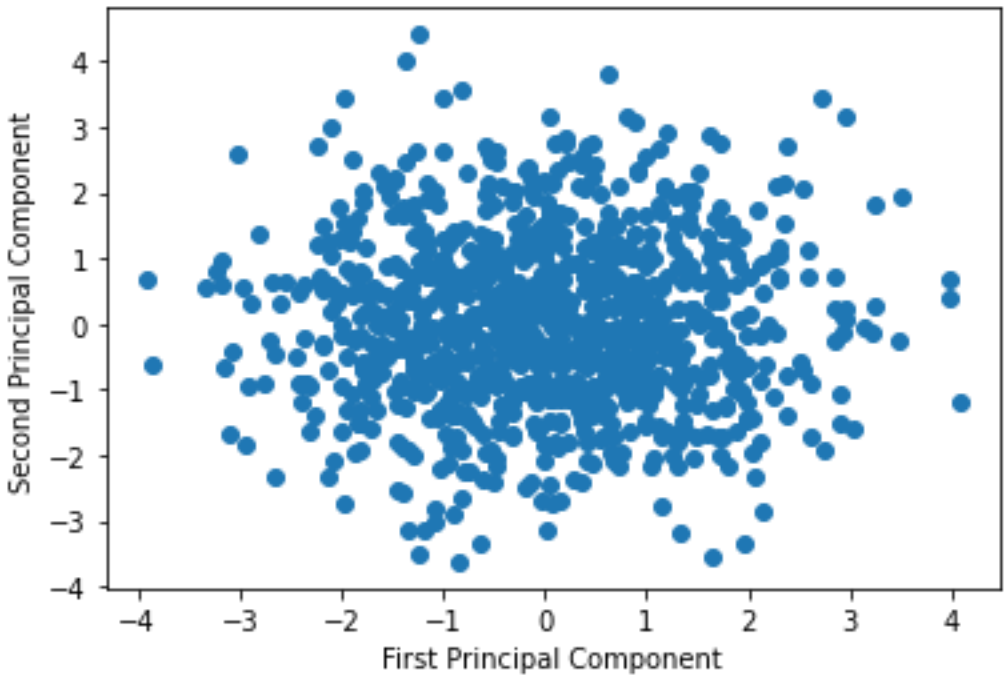
c) import matplotlib.pyplot as plt
cum_sum_exp = np.cumsum(S)
plt.bar(range(0,len(S)), S, alpha=0.5, align='center',
label='Individual explained variance')

```
plt.step(range(0,len(cum_sum_exp)), cum_sum_exp,
where='mid',label='Cumulative explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal component index')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

here is the plot:



d)
```
data_projected = data @ Vt[:2].T
plt.scatter(data_projected[:, 0], data_projected[:, 1])
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.show()
```



Question 5: k-NN and Linear Reg

1. In the k-NN regression, the target value for a given data point is calculated as the average of the target values of its k nearest neighbors in the training set. The algorithm first selects the k nearest neighbors of the test data point using a distance metric, such as Euclidean distance. The target values of these k nearest neighbors are then averaged to obtain the predicted target value for the test data point.
2. Linear Regression in Depth

   a)

$$argmin\ \frac{1}{n} * \ \Sigma_{\{i=1\}}^{n}\left(y_i - \ (w^T * \ x_i)\right)^2$$

   b) Here is the formulation:

$$p\left(y(i)\middle|x(i)\right) = \frac{1}{sqrt(2\pi\sigma^2)} * \exp\left(-\frac{\left(y(i) - \ w^T x(i)\right)^2}{(2\sigma^2)}\right)$$

$$L(y,w) = \prod_{\{i=1\}}^{n}\left(\frac{1}{sqrt(2\pi\sigma^2)}\right) * \exp\left(-\frac{\left(y(i) - \ wTx(i)\right)^2}{2\sigma^2}\right)$$

For this step we need to get the argmax of the L(y, w) in respect to w:
As we know since the log is an increasing fuction we can get argmax of the log(L) instead of L:

$$argmax\ L(y,w) = argmax(Log^{L(y,w)})$$

$$\log\left(L(y,w)\right) = \log\left(\prod_{\{i=1\}}^{n}\left(\frac{1}{sqrt(2\pi\sigma^2)}\right) * \exp\left(-\frac{\left(y(i) - \ w^T x(i)\right)^2}{(2\sigma^2)}\right)\right)$$

$$\log\big(L(y,w)\big) = \sum_{\{i=1\}}^{n} \log\left(\left(\frac{1}{sqrt(2\pi\sigma^2)}\right) * \exp\left(-\frac{\big(y(i) - w^T x(i)\big)^2}{(2\sigma^2)}\right)\right)$$

$$\log\big(L(y,w)\big) = -\left(\frac{n}{2}\right) * \log(2\pi\sigma^2) - \sum_{\{i=1\}}^{n} \frac{\big(y(i) - w^T x(i)\big)^2}{(2\sigma^2)}$$

Since we can delete the constant part when getting argmax we have:

$$argmax\ L(y,w) = argmax\ -\sum_{\{i=1\}}^{n} \frac{\big(y(i) - w^T x(i)\big)^2}{(2\sigma^2)}$$

Which is equal to:

$$argmin \sum_{\{i=1\}}^{n} \frac{\big(y(i) - w^T x(i)\big)^2}{(2\sigma^2)}$$

And this is exactly what we have found in the ERM interpretation.

$$L(y,w) = |Xw - y|^2 = (Xw - y)^T (Xw - y)$$

$$L(y,w) = w^T x^T xW - 2w^T x^T y + y^T y$$

$$L(y,w) = w^T W|x|^2 - 2w^T x^T y + y^T y$$

$$\nabla L(y,w) = 2X^T Xw - 2X^T y = 0$$

$$X^T Xw = X^T y$$

$$w = (X^T X)^{-1} X^T y$$

c) We know that y − #y is orthogonal to all vectors in the subspace. So, we will have:

$$x^T(y - \hat{y}) = 0$$
$$x^T(y - x\hat{w}) = 0$$
$$x^T x\hat{w} = x^T y$$
$$\hat{w} = (x^T x)^{-1} x^T y$$