ECE 479 ICC: IoT and Cognitive Computing

Spring 2023, Homework

Due on Feb. 7, 2023 at 11:59:59 PM

Question 1: Edge Computing and Sensors (25 pts)

1. List two advantages and disadvantages of using Cloud Computing or Edge Computing for an IoT system. **[8 pts]**
   - Cloud Computing:
     - Advantages:
       (a)
       (b)
     - Disadvantages:
       (a)
       (b)
   - Edge Computing:
     - Advantages:
       (a)
       (b)
     - Disadvantages:
       (a)
       (b)

2. For the following applications, which paradigm (Cloud Computing or Edge Computing) is more suitable? Briefly explain why. **[6 pts]**
   - Health data collected by an smartwatch to track your daily activities.
   - Temperature sensors are placed inside a refrigerated storage container to regulate the temperature during the shipping process.
   - License plate readers at toll plazas.

- Medical wearable devices that detect when you fall.

3. Wireless Sensors Network (WSN) Aggregation Strategy. **[11 pts]**

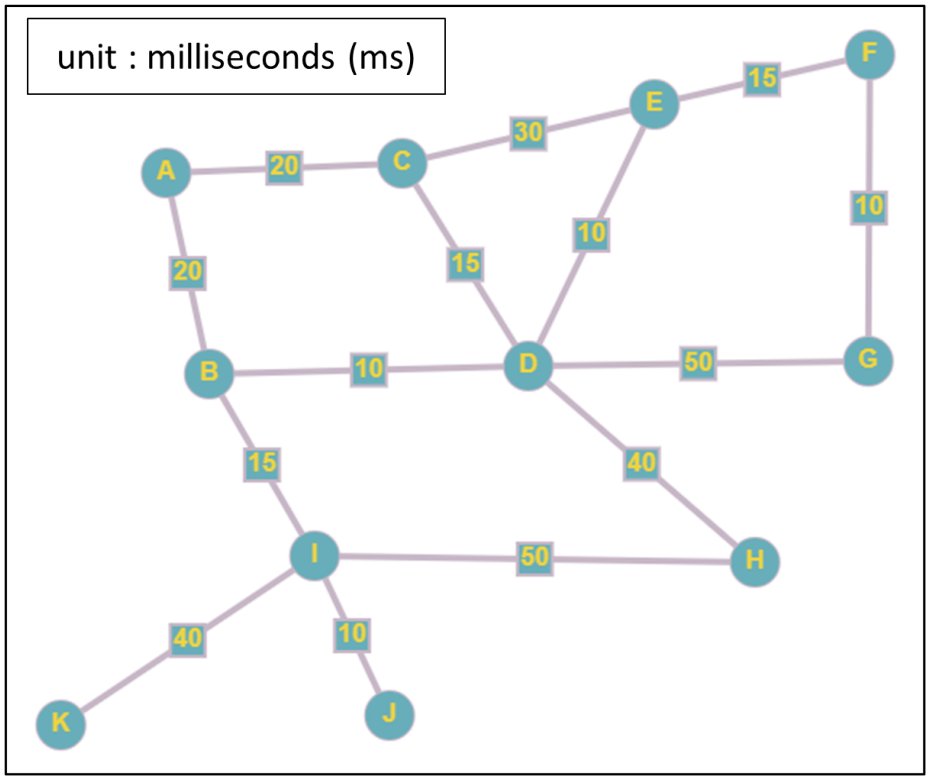

Figure 1: WSN topology

All nodes except A are sensors, and node A is the sink node. The numbers on the edges are the corresponding transmission latency. To find out the shortest path between two connected nodes in a graph, you many need the help of the Dijkstra's algorithm.

We want to collect data from sensors K, E and G and transmit back to sink node A. Follow the same methodology we introduced in the lectures, please propose a plan, and calculate minimum **total** latency and the number of operations required in the transmission.

Assume each aggregation requires 5 milliseconds (ms) to complete. Justify your answer and fill in the following table. (You may not need to fill in all columns.)

# of Operations	K	E	G		
Transmission					
Aggregation					
Total					

Table 1: Wireless Sensors Network Traversing

Question 2: Basic Python and NumPy programming (15 pts)

1. **Python One-Liners: Basics [6 pts]**
   (a) Write down **a single line** Python expression that calculates $\sum_{n=0}^{i} n^2$ using only Python built-in functions. **[1 pts]**
   (b) Write down **a single line** of Python code to sum the 0, 3, 6, 9,...,(3n)th elements from a list A using only Python built-in functions. **[1 pts]**
   (c) Write down **a single line** of Python code to sum the 2, 5, 8, 11, (3n+2)th elements from a list A using only Python built-in functions. **[1 pts]**
   (d) Write down **a single line** of Python code to reverse given list A and save it to list B. **[1 pts]**
   (e) Given two lists of numbers in Python, A and B, write down **a single line** of Python code to construct a new list of pairs of corresponding elements in two list. For example, A = [1, 2, 3], B = ['a', 's', 'd', 'f'], then your code should give [(1, 'a'), (2, 's'), (3, 'd')]. **[1 pts]**
   (f) Given two lists of numbers in Python, A and B, write down **a single line** of Python code to construct a new list of elements in A that appear in B as well. For example, A = [1, 2, 4, 4, 2, 1], B = [1, 4], then your code should give [1, 4, 4, 1] **[1 pts]**
2. **More one-liners: List and String Operations [4 pts]**
   Use the following list and string methods:
   `append(), count(), extend(), insert(), join(), remove(), split()`
   Fill in the blank (parenthesis) to output the expected results. The comments are what you should expect after you have run the following line of code. You should only use one method in each blank.

```
# split string s0 by space and put into a list s0_list
# output: ['Oppenheimer', 'drinks', 'some', 'coffee', 'coffee',
   'coffee', 'coffee']
s0 = "Oppenheimer drinks some coffee coffee coffee coffee"
s0_list = (                                          )
```

```
# count the occurrence of element 'coffee'
# output: 4
count = (                                          )

# keep only the first four elements in the list
# output: ['Oppenheimer', 'drinks', 'some', 'coffee']
s0_list = (                                        )

# take out element 'some' from the list
# output: ['Oppenheimer', 'drinks', 'coffee']
s0_list.(                                         )

# put element 'much' in the list between 'drinks' and 'coffee'
# output: ['Oppenheimer', 'drinks', 'much', 'coffee']
s0_list.(                                         )

# put element 'everyday' at the end of the list
# output: ['Oppenheimer', 'drinks', 'much', 'coffee', 'everyday
    ']
s0_list.(                                         )

# split string s1 by space and put into a list
# output: ['in', 'the', 'afternoon']
s1 = "in the afternoon"
s1_list = (                                        )

# put s1_list at the end of s0_list
# output: ['Oppenheimer', 'drinks', 'much', 'coffee', 'everyday
    ', 'in', 'the', 'afternoon']
s0_list.(                                         )

# make a string from the list, connected with a space ' '
# output: Oppenheimer drinks much coffee everyday in the
    afternoon
s = (                                              )
```

3. **NumPy Slicing [2 pts]**

The **shape** of a NumPy array *tmp* is (5, 3, 4, 2), which corresponds to dimensions $[i, j, k, l]$.

```
tmp = tmp[2:4, :-1, ::2, :1]
```

What is the shape of the NumPy array *tmp* and which part of the original NumPy array *tmp* has been extracted after this slicing?

Please describe it in terms of i, j, k, l dimension. (for example : row 0 and row 1 of dimension i and .... of dimension j and ...)

4. **NumPy Operations [3 pts]**
We have `A = np.array([1, 2, 3, 4, 5, 6])` and a mystery NumPy array `B`. Printing `A*B` gives the output of `[2 14 9 20 30 24]`. What is this mystery NumPy array `B`?

Now suppose that NumPy array `A1` and `B1` are transformed from A and B using NumPy array manipulation functions.

Please find out a way to perform these transformations from `A` and `B`, using NumPy array operations, such that

```
print(np.dot(A1, B1).ravel())
```

gives the output of [10 20 9 24 46 23 38 72 37]

Question 3: Machine learning concepts (20 pts)

1. **Supervised vs. Unsupervised Learning**
   We can roughly divide the machine learning algorithms into two categories: supervised learning and unsupervised learning.

   (a) In two sentences, explain a key difference between the supervised and unsupervised learning. **[3 pts]**

   (b) Among the following tasks, decide whether you should use supervised learning or unsupervised learning and briefly explain why.**[4 pts]**

   - Identify whether a person belongs to this company, using facial recognition.
   - Find how many different customer groups are visiting a shopping website.
   - Approximate the daily schedule of a certain person.
   - Predict the market price of a house in the neighborhood, using historical data.
   - Enhance the details of a blurry image with machine learning.

   (c) We have worked with k-Nearest Neighbor algorithm in lab 1 to classify flowers. Is this supervised learning or unsupervised learning? Why or why not? How about K-Means?**[5 pts]**

2. **Fitting the Data**
   Suppose that we have collected some data points for training and plotted them in the scattered plots, shown below. To find the trend of these data points, we constructed three regression models. Later, we want to use one of these models to describe the incoming unknown data points. Observe the following three graphs and answer the questions.

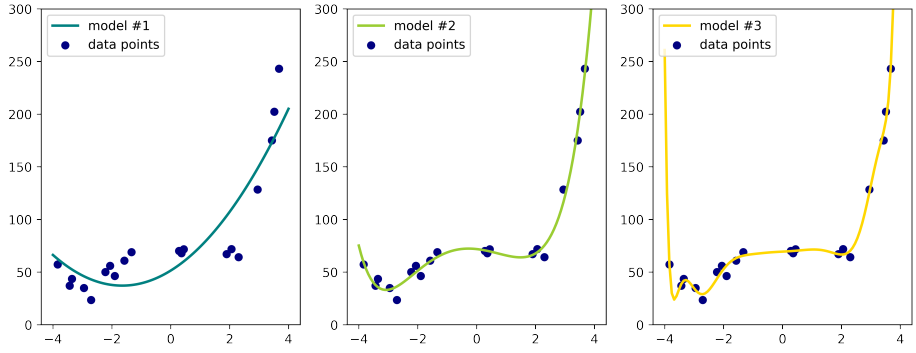

Figure 2: Three Different Models

(a) Among the three models, which one do you think is the best? Why do you choose this model? **[2 pts]**

(b) What are the potential issues with the other two? Please explain with two to three sentences. **[2 pts]**

3. **Training and Validation**
When developing a machine learning model, we usually need to evaluate the model's accuracy before testing it on real-world data. Since we only have access to the training dataset, we often have to split it into two parts, one for training and the other for validation.

(a) In about three sentences, describe a simple scenario where you need to use training/testing split and why. **[2 pts]**

(b) Why do we need validation/cross-validation if testing set is enough to evaluate the real-world performance of a model? That is, why can't we tune the model based on testing accuracy? **[2 pts]**

Question 4: PCA and Preprocessing of data (20 pts)

1. **Curse of dimensionality[2 pts]**
Briefly explain "the curse of dimensionality". Why high dimensional data is a "curse"? What makes it difficult to model high-dimensional data?

2. **Statistical interpretation of PCA[2 pts]**
In the lectures, you have learned the statistical interpretation of PCA. Briefly describe it below.

3. **Linear algebra interpretation of PCA[4 pts]**
PCA can also be interpreted as: finding a subspace $\mathbb{S}$, represented with a set of orthonormal basis $v_1, ..., v_k$, such that the sum of the squared difference between the original input vectors and the projection onto the subspace $\mathbb{S}$ is minimized. For this question, we define the matrix $V \in \mathbb{R}^{d \times k}$ whose column vectors are the orthonormal basis $v_1, ..., v_k$, and the matrix $X \in \mathbb{R}^{n \times d}$ whose rows are the input vectors $x_1, ..., x_n$.

You can refer to this note for the details of projecting onto a subspace. Notice that the projection of the input vectors onto a space is given by PX^T in our case because the rows of X are the vectors to be projected (P is the projection matrix). Also, notice that $V^TV = I$, since all the column vectors of V are orthogonal.

Given the description above, derive the objective function in terms of X, V, and

the squared Frobenius norm (the sum of squares of all elements in a matrix), also simplify the equation as much as you can.

4. **PCA with NumPy [12 pts]**
   Consider a dataset with 1000 samples and 100 features. The samples are stored in a numpy array called data. Let's say, `data = np.random.randn(1000 ,100)`

   (a) **[3 pts]** Standardize the data by subtracting the mean and dividing by the standard deviation for each feature. Please use `np.mean` and `np.std`.

   (b) **[3 pts]** Perform PCA on the standardized data using NumPy's `numpy.linalg.svd()` function to obtain the principal components and explain variance. Explain the returned values of the function.

   (c) **[3 pts]** Plot the explained variance as a function of the number of principal components. You can refer to the article for more information.

   (d) **[3 pts]** Project the data onto the first two principal components and plot the resulting 2D data points.

Question 5: kNN and Linear Regression (20 pts)

1. **Simple regression with kNN [2 pts]**
   In your own words, describe how the kNN algorithm calculates the regression. You can assume that all the dependent and independent variables are numerical.

2. **Linear regression in greater depth**
   Linear regression, as the very basic of statistical learning, is one of the simplest, yet also an extremely effective algorithm. Interestingly, it can be interpreted from 3 different perspectives:

   - Machine learning/Empirical Risk Minimization (ERM)
   - Statistics/Maximum Likelihood Estimation (MLE)
   - Geometric/Linear algebra

   So in this questions we will guide you to have an in-depth view of the three perspectives.
   Note: in this question, we will omit the constant offset for simplicity, adding the offset will be the same as having one additional feature whose value is 1 for all samples.

   (a) **Machine learning/Empirical loss minimization [6 pts]**:
   The first interpretation of this problem is straightforward: we simply want a linear function that can produce the minimal error of a given form, and

for linear regression, we usually pick the mean squared error (MSE). To begin with, we denote the set of parameters (weights) with $\boldsymbol{w}$ and the features of the i-th sample with $\boldsymbol{x}_{(i)}$. So the prediction for the i-th sample is:

$$\hat{y}^{(i)} = \boldsymbol{w}^T \boldsymbol{x}^i$$

Given the information above, write down the equation of the mean squared error (MSE) to be minimized, the first part of the equation is given to you.

$$\underset{\boldsymbol{w}}{\text{argmin}} \_\_\_\_\_$$

Such an approach is called *Empirical Risk Minimization (ERM)* in the machine learning world. We will derive how to obtain the $\hat{\boldsymbol{w}}$ that minimizes the loss in the next part.

(b) **Statistical/Probabilistic interpretation [6 pts]**:
We can also interpret the logistic regression as a probabilistic problem, in which the independent variables $y^{(i)}$ are assumed to obey a Gaussian distribution $y^{(i)} \sim G(\boldsymbol{w}^T \boldsymbol{x}^{(i)}, \sigma^2)$. In other words, the mean (μ) of the Gaussian distribution for each feature is controlled by the input values and the parameter. Given this information, fill in the gaps in the following equation for the probability:

$$p(y^{(i)}|\boldsymbol{x}^{(i)}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(\_\_\_\_\_)^2\right)$$

Still, our goal is to find the best set of parameters $\hat{\boldsymbol{w}}$. In statistics, one of the most commonly used method to find the parameters of a distribution is the *Maximum Likelihood Estimator (MLE)*, and it is calculated by taking the product of the conditional probability of all the points. Recall or research for the equation of MLE and write down below:

$$L(\boldsymbol{y}, \boldsymbol{w}) = \_\_\_\_\_$$

Next, substitute in all the equations and do the standard MLE steps to find the expression of the $\boldsymbol{w}$ that maximizes the likelihood, and show your steps. Simplify the right hand side to only contains terms related to $\boldsymbol{x}$, $\boldsymbol{w}$ and y if you need. (Hint: constants does not affect the min or max function) As you will see, this is exactly what we have in the ERM interpretation.

Now the problem becomes "how do we minimize this value?" To do that, we can directly differentiate the loss function and set the derivative to 0 as usual. Notice that, the loss function can be alternatively expressed in matrix form. Consider forming a matrix $\boldsymbol{X}$ as below, where each row represents a sample in the training dataset. Recall that a prediction is made by multiplying each feature/independent variable in the sample with a fitted parameter value, which can be seen as a matrix-vector multiplication.

$$L(\boldsymbol{y}, \boldsymbol{w}) = |\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}|^2$$

Recall, from your linear algebra course, that $|\boldsymbol{x}|^2 = \boldsymbol{x}^T\boldsymbol{x}$. So, expand the above equation using that and write down the result below: (your final result should not have $\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$ in it)

Now, we can apply the following matrix differentiation rules:

$$\frac{\partial \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{x}$$
$$\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{A}^T$$

For more derivations, you can refer to this great reference. So next, assuming that X is invertible, using these rules, write down the final derivation of the expression of $\boldsymbol{w}$ in terms of $\boldsymbol{X}$ and $\boldsymbol{y}$. (Hint: the final result is the Ordinary Least Squares (OLS), which is $\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$)

(c) **Geometrical explanation [6 pts]**:
The linear regression can also be explained as a geometry/linear algebra problem. From this perspective, $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}$ has an alternative interpretation: a linear combination of the column vectors of $\boldsymbol{X}$.

In this interpretation, the vector of truth values $\boldsymbol{y}$, can be seen as a vector that points out from the hyper-plane. So, minimizing the error becomes minimizing the distance between the vector constructed with the vectors on the plane $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}$ and the ground truth vector $\boldsymbol{y}$, as demonstrated in the following figure.

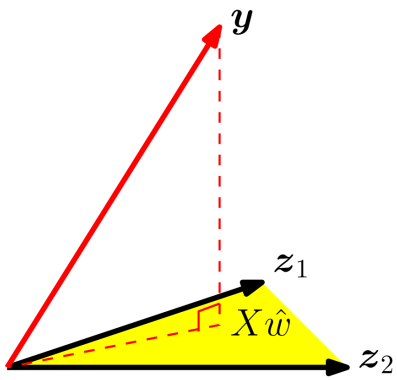

Figure 3: Demonstration of the least squares (Source: CS 446)

In geometry, this is the same as minimizing the magnitude of the vector $\boldsymbol{y} - \hat{\boldsymbol{y}}$. From pure geometry knowledge, we know that such a vector is shortest when it is perpendicular to the surface, in which case it will be orthogonal to all the vectors in that plane. Using this relationship, we can then derive the analytical expression of $\hat{\boldsymbol{w}}$. Please write down the steps below and your final result should be exactly the same as what you get in part (b).