Master's and Ph.D. Qualification Exam

Graduate School of Data Science

2021/06/22

Area 1. Probability and Statistics

You are given 2 questions in this area. Please answer **all of them**.

1. Suppose you want to estimate the number of people who were previously infected by COVID in the US using an antibody test with a budget of testing 2,000 individuals. It is known that 60% of people with COVID have a fever. Overall 10% of people had a history of fever during the last one year. So, researchers carried out a study by randomly selecting half of the sample (1,000) with a history of fever and half of the sample (1,000) without fever.

(a) Suppose that the true rate of previous infection is 9%. On average, how many people would be previously infected among 1,000 people with a history of fever? How many among 1,000 people without a history of fever?

(b) Now researchers observed that 550 and 40 people had COVID antibodies (so previously infected by COVID) with and without fever, respectively. Estimate the rate of people with COVID antibody in the US.

(c) Researchers want to obtain the confidence of the results. Calculate the variance of the rate estimator in b) and 95% confidence interval. For 95% interval, you can provide a formula without the final number (Note: Z-values are $Z_{0.9} = 1.28, Z_{0.95} = 1.64, Z_{0.975} = 1.95$).

(d) Suppose that the true rate of infection is 9% as noted in a). Compared to the universal random sampling regardless of the fever, do you think this design is better (Yes/No)? Please justify your answer.

2. Briefly explain what bias-variance trade-off is, and state a possible solution to balance between bias and variance. Please justify your answer for your proposed solution.

Area 2. Computing

You are given 2 questions in this area. Please answer **all of them**.

1. Briefly describe what the following method `func` does.

```
def func(linked_list):
  # linked_list: the head node of a sorted linked list.
  # Each node in this linked list has two attributes, data and next.

  it = linked_list
  while it:
    item = it.next
    while item and item.data == it.data:
      item = item.next
    it.next = item
    it = item
  return linked_list
```

2. Consider the following two tables:

Column name	Description	Data type	Distinct values
`orderid`	Order ID (primary key)	INT(4)	830
`productid`	Product ID of the order	INT(4)	77
`quantity`	Order quantity	INT(4)	55
`discount`	Price discount of the order (ex: 0.15 means 15% discount)	FLOAT(8)	7

Table 1: Schema for table `Orders`

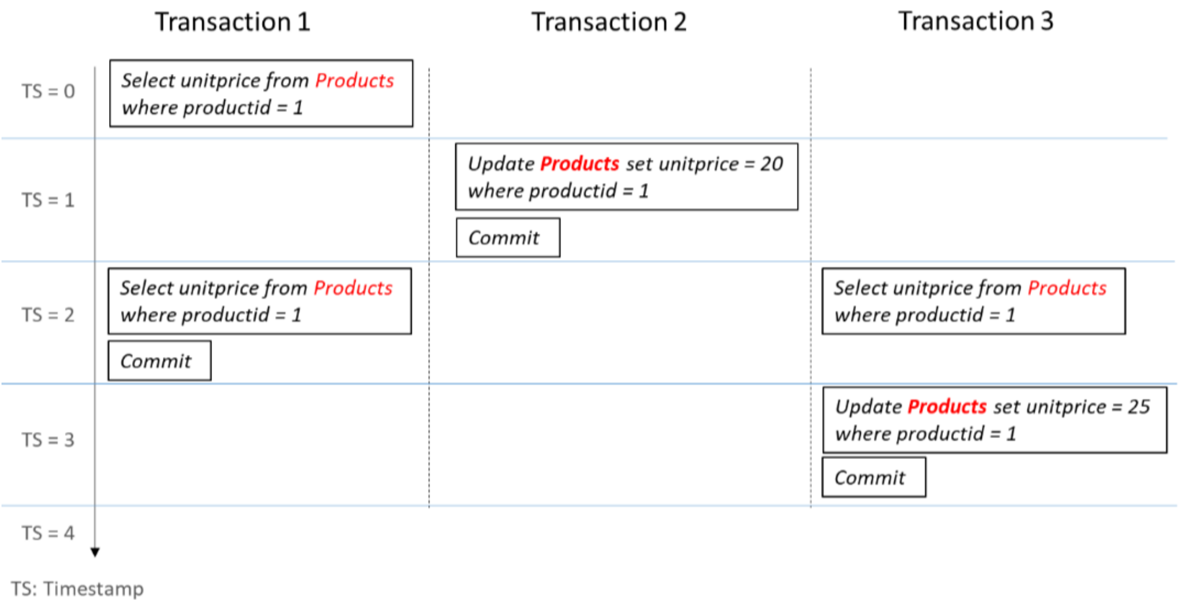
Column name	Description	Data type	Distinct values
`productid`	Product ID (primary key)	INT(4)	77
`supplierid`	Supplier ID of the product	INT(4)	29
`categoryid`	Category ID of the product	INT(4)	8
`unitprice`	Unit price of the product	FLOAT(8)	62

Table 2: Schema for table `Products`

(a) Write an SQL query to compute the total revenue per each supplier, sorted in decreasing order. Revenue for each order is computed as "`Orders.quantity * Products.unitprice * (1 - Orders.discount)`".

(b) Suppose we store `Orders` and `Products` tables in column table form using dictionary encoding. Compute the memory size needed for each table.

Now, consider the following transactions on the `Products` table.


Transaction 1
Transaction 2
Transaction 3
TS = 0
Select unitprice from Products where productid = 1
TS = 1
Update Products set unitprice = 20 where productid = 1
Commit
TS = 2
Select unitprice from Products where productid = 1
Commit
Select unitprice from Products where productid = 1
TS = 3
Update Products set unitprice = 25 where productid = 1
Commit
TS = 4
TS: Timestamp

Initially (`TS` = 0), `unitprice` is 18.0 and 19.0 for `productid` = 1 and 2, respectively. Given this information, answer to the following questions.

(c) Assuming MVCC, explain the values read from each `select` query using version space diagrams.

(d) Assuming that the DBMS uses undo logging only, write the log at `TS` = 4.

(e) Assuming that we have B+ tree index for the `productid` column in the table `Orders` and `Products`, describe a linear time algorithm to join the two tables.

Area 3. Machine Learning

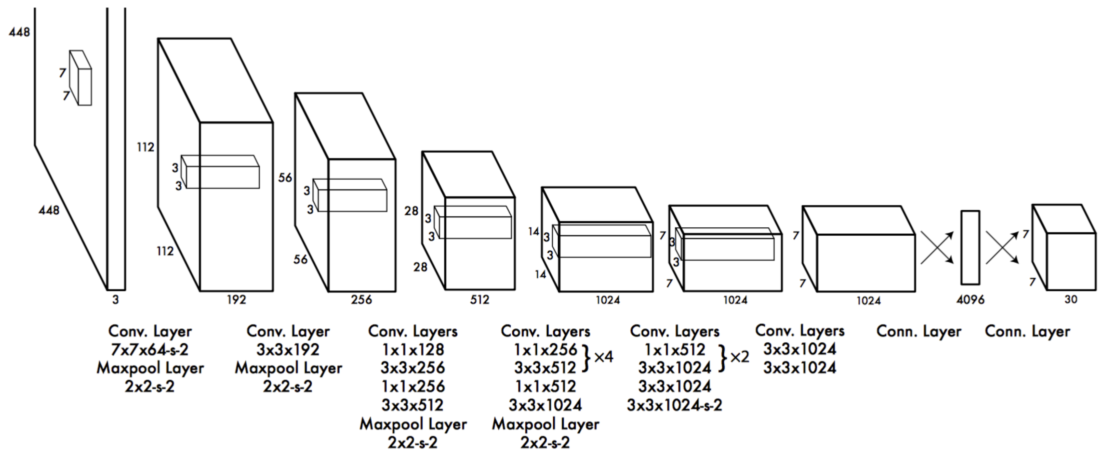
You are given 3 questions in this area. Answer **two questions of your choice**. Clearly mark your chosen questions. (Otherwise, we will grade the first two questions you submit.)

1. Answer the following questions about convolutional neural networks.

(a) Is a convolutional layer a special case of a fully-connected layer? Choose yes or no, and justify your answer.

(b) Is a fully-connected layer a special case of a convolutional layer? Choose yes or no, and justify your answer.

(c) In image models, 1×1 convolutions are widely used, even though the filters do not look at neighboring pixels. The model shown below is an object detection model called YOLO, an example using 1×1 convolutoinal layers. Explain why 1×1 conv is used in practice even if it does not learn spatial locality.



2. Ensemble techniques are widely used to improve classification performance. Answer to the following questions about ensemble techniques.

(a) Explain what bagging and boosting are, and how they can improve the classification results over decision trees.

(b) Compare pros and cons of Random Forest and Ada-boost (or XG-boost) from at least two aspects.

3. Stochastic gradient descent (SGD) and its variants are widely used for optimization in machine learning. What is the difference between "gradient descent" and "SGD"? How does *stochasticity* in SGD help in optimization?

Programming

You are given 3 questions in this section. Answer **two questions of your choice**. Clearly mark your chosen questions. (Otherwise, we will grade the first two questions you submit.)

Instructions to submit your solutions:

1. The files `QE_prob1.py`, `QE_prob2.py`, and `QE_prob3.py` contain your solution to problems 1, 2, and 3, respectively.
2. Remove any debugging or logging code before you submit. It may disturb the automatic grading process, and as a result, you will likely get a lower score.
3. Compress the three files `QE_prob1.py`, `QE_prob2.py`, and `QE_prob3.py` to a single submission `file 20XX_XXXXX.zip` (`20XX_XXXXX` is your SNU student id, *e.g.*, 2020_12345.zip). The submission file should contain at most three files: `QE_prob1.py`, `QE_prob2.py`, and `QE_prob3.py`.
4. Send the submission file to `gsds_qe@aces.snu.ac.kr` from your SNU email account (if it is not an SNU email account, we will not accept your solution). The title of the submission email should be `[QE] 20XX-XXXXX` (*e.g.*, `[QE] 2020-12345`).
5. Make sure that the attached file is easily downloadable from the email message. We will not accept any submission that requires third-party tools or storages (*e.g.*, Google Drive).

Note: You may use the Internet for API search, but communication with other people in any matter is strictly prohibited. Violation to this will be considered as academic misconduct.

1. Given an integer `array` of size N. Implement a method that returns the smallest positive integer which is not in that array. To get full credit, your implementation needs to run in $O(N)$. (That is, the run time *linearly* increases *only* to the array size, regardless of the element values.) Do not use the built-in `sort` method. If needed, build it on your own.

- Example 1) array = [4, 7, -1, 1, 2] $\rightarrow$ return: 3
- Example 2) array = [100, 101, 102] $\rightarrow$ return: 1
- Example 3) array = [3, 2, 1, 0, -1] $\rightarrow$ return: 4

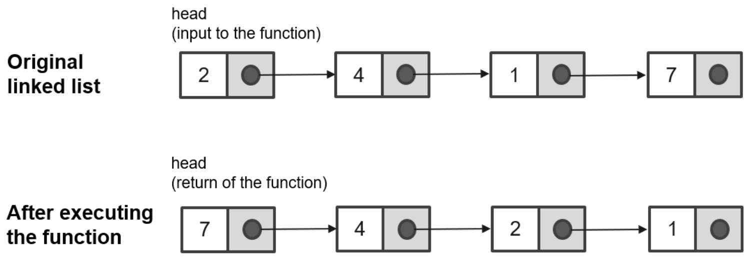
Please implement the `smallest_pos_int(array)` in the file `QE_prob1.py`.

2. There is a singly linked list where each node is `LinkedNode` defined below.

```
class LinkedNode:
  def __init__(self, x):
    self.val = x
    self.next = None
```

Given the head (first node) of the linked list, sort the linked list in descending order (`val` element) and return the head of the sorted linked list. To get full credit, your implementation needs to run in $O(N \log N)$ with $O(1)$ memory space. Do not use the built-in `sort` method. If needed, build it on your own.

- Example)



Please implement the following `sortingLL(head)` in the file `QE_prob2.py`.

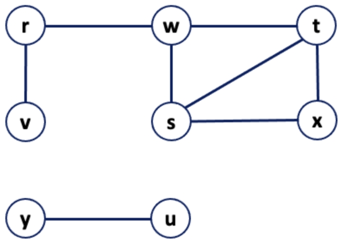
```
def sortingLL(head: LinkedNode) -> LinkedNode:
  # Your Code
```

3. In this problem, you will implement a function that finds out the lexicographically smallest path between two nodes in a graph. A node in an undirected graph is defined as follows (do not modify the node definition in your solution):

```
class GNode:
  def __init__(self, id, color="W";, d=-1, f=-1, p=None):
    self.id = id          # id is a string
    self.color = color    # color (status) of node
    self.d = d            # discover time of node
    self.f = f            # finish time of node
    self.parent = p       # predecessor time of node

  def __str__(self):
    return self.id
```

Consider an adjacency list implementation of an undirected graph using a dictionary. You should implement a function `LexSmallest(G, u, v)` that takes an undirected graph `G` and two vertices `u` and `v` as arguments. It returns the lexicographically smallest path between `u` and `v`. The list contains the id of the nodes on the path (*i.e.*, it is a list of strings). If there is no path between `u` and `v`, `LexSmalles(G, u, v)` returns an empty list. For example, consider the following graph `G`:



You can create `G` in an adjacency list implementation using a dictionary:

```
>> r, s, t, u = GNode('r'), GNode('s'), GNode('t'), GNode('u')
>> v, w, x, y = GNode('v'), GNode('w'), GNode('x'), GNode('y')
>> G = dict()
>> G[r], G[w], G[t], G[u] = [w, v], [s, r, t], [s, x, w], [y]
>> G[v], G[s], G[x], G[y] = [r], [w, t, x], [s, t], [u]
```

Then some behaviors of `LexSmallest()` are as follows:

```
>> LexSmallest(G, t, v)
['t', 's', 'w', 'r', 'v']
>> LexSmallest (G, u, u)
['u']
>> LexSmallest(G, w, y)
[]
```

The lexicographic order (also known as dictionary order) is a generalization of the alphabetical order of the dictionaries to sequences of ordered symbols. For two lists of strings `P1` and `P2`, `P1` is lexicographically smaller than `P2` if and only if there exists an integer k such that `P1[0] = P2[0]`, `P1[1] = P2[1]`, ..., `P1[k-1] = P2[k-1]`, and `P1[k] < P2[k]`. In the example graph `G`, the lexicographically smallest path between `t` and `v` is `[‘t’, ‘s’, ‘w’, ‘r’, ‘v’]`. The returned list must include every vertex in the lexicographically smallest path. Your solution should be based on the Depth-First Search (DFS) algorithm.

2-Day Project

You are given 2 questions in this area. Answer 1 question of your choice.

1. Seoul Air Prediction

Seoul Air Quality dataset[1] was used in [1, 2, 3][2], including Seoul air quality data from 2008 to 2018. Air quality is impacted by many factors such as traffic volume, neighboring area situations, weather, seasonal information, and other economic activities. Many works have addressed the relationship between the air quality level and other factors via numerous modeling approaches. For instance, during the Chuseok holidays, the air quality index tends to get better, while it is serious during weekdays, especially with foggy weather conditions or in the yellow dust season. You may refer to [1, 2, 3] for more information on how researchers used this dataset in their works.

Columns	Meaning
Datetime	Date and time
District	District code 0-25 (Code 0 represents the average value of all 25 districts in Seoul). Other districts are identified from 1 to 25.
PM10_CONC	PM10 concentration ($\mu g/m^3$)
PM2_5_CONC	PM2.5 concentration ($\mu g/m^3$)
O3	Ozone concentration ($\mu g/m^3$)
NO2	NO2 concentration ($\mu g/m^3$)
CO	CO concentration ($\mu g/m^3$)
SO2	SO2 concentration ($\mu g/m^3$)
PM10_AQI	PM10 AQI Index according to US Standard AQI Index
PM2_5_AQI	PM2.5 AQI Index according to US Standard AQI Index

Table 3: Seoul Air Quality Dataset

(a) **Additional Data Collection.** Of course, it is okay to treat the problem as a pure time-series problem. After that, you can apply traditional methods like ARIMA or SVR to solve the problem. However, these methods are only appropriate for data that have repeated patterns. In many problems, especially Air Quality forecasting, more relevant data help improve the accuracy of prediction models. Therefore, you may collect additional weather and holiday information. In [1], the author clearly explains how different data sources are collected. If you can collect more than two data sources and use them in your model, you get full credits. Please specify what additional datasets you used.

(b) **Data Cleansing & Data Loading.** Describe how you preprocess the data into a format you use for training.

(c) **Learning Algorithm.** Describe your learning algorithm and how it is implemented.

(d) **Evaluation Metric.** Define your evaluation metric and explain why it is appropriate for the problem.

(e) **Experimental Settings.** Clearly specify experimental settings and how you split the dataset into train/val/test sets.

(f) **Result and Discussion.** Present your work at the oral exam, including (but not limited to) problem setting, hypotheses, how you collect and preprocess data, description and justification of your learning algorithm, prediction results, interesting observations or insights learned, and potential future work.

[1] `https://cleanair.seoul.go.kr/airquality/localAvg`
[2] `https://github.com/alexbui91/Air-Quality-Prediction-Datasets`

2. **New COVID Infection Prediction**

Predicting future COVID19 infection is of a major importance in public health. The rate of infection can be affected by many factors including weather/temperature, government containment policy, economic policy, etc. Recently vaccination becomes an important factor. In this project, students need to use many different sources of data to build a model to predict short term (next 10 day) and more longer term (next 1 month) prediction of the number of new COVID cases. The following data can be used.

- COVID new case statistics
  `https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-data.csv`
- Government response data
  `https://www.bsg.ox.ac.uk/research/research-projects/covid-19-government-response-tracker#data`
  `https://github.com/OxCGRT/covid-policy-tracker`
- Vaccination data
  `https://github.com/owid/covid-19-data/tree/master/public/data/vaccinations/country_data`

(a) **Country to build the model.** To build a model to predict COVID infection for two countries: Korea and US

(b) **Additional Data Collection.** In addition to the data listed, you can also used additional data to fit the model. When you use additional data, please specify the datasets.

(c) **Data Cleansing & Data Loading.** Describe how you preprocess the data into a format you use for training.

(d) **Learning Algorithm.** Describe your learning algorithm and how it is implemented.

(e) **Evaluation Metric.** Define your evaluation metric and explain why it is appropriate for the problem.

(f) **Experimental Settings.** Clearly specify experimental settings. To find the model, you may need to split the dataset into train/val/test sets. Please specify how this can be done.

(g) **Result and Discussion.** Present your work at the oral exam, including (but not limited to) problem setting, hypotheses, how you collect and preprocess data, description and justification of your learning algorithm, prediction results, interesting observations or insights learned, and potential future work.

References

[1] Tien-Cuong Bui, Joonyoung Kim, Taewoo Kang, Donghyeon Lee, Junyoung Choi, Insoon Yang, Kyomin Jung, and Sang Kyun Cha. STAR: Spatio-temporal prediction of air quality using a multimodal approach. In *Proc. of SAI Intelligent Systems Conference*, pages 389–406, 2020.

[2] Sookyung Kim, Jungmin M Lee, Jiwoo Lee, and Jihoon Seo. Deep-dust: Predicting concentrations of fine dust in seoul using LSTM. *Climate Informatics*, 2019.

[3] Van-Duc Le, Tien-Cuong Bui, and Sang-Kyun Cha. Spatiotemporal deep learning model for citywide air pollution interpolation and prediction. In *IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 55–62, 2020.