COMP 379-001/479-001 Machine Learning - Spring 2025
Instructor: Daniel Moreira (dmoreira1@luc.edu)

Student (Printed): ________________________________ **(Signature):** _____________________

Midterm Exam - 03/13/2025

Suppose you work for a software development company whose expertise is machine learning-based systems. One of the company's projects is the development of a flash flood alert system for a city situated along a river bank. Given the city's susceptibility to frequent flooding, the system's objective is to issue timely three-level severity alerts (namely, yellow, orange, and red) to the citizens through SMS, letting them evacuate before major issues occur.

The system will replace an existing three-level alert protocol, which is manually triggered by human observers after inspecting several instruments, such as rain gauges, river level gauges, river speed monitors, thermometers, anemometers, and barometers. The observers decide the need for an alert and its severity based on their own judgment of the expected river level for the next hour or so, considering all the available instruments' information. The history of alerts with the values of the instruments, date, and time have been collected for the past 50 years.

One of your developer peers insists that one can implement the system rules to define the need for and severity of the alerts by simply interviewing the human observers and figuring out what instruments' thresholds they adopt to decide and take action. After interviewing the currently hired three observers and dealing with some confusion – the three folks do neither agree on the adopted thresholds nor their precedence – the developer came up with a first set of rules (a chain of if-then-else structures), which he is willing to maintain as the system development and test moves forward. To make things worse, during its 50-year existence, the manual protocol had several observers who are not available anymore for interviewing.

Table 1 summarizes the data collected in the past 50 years. Algorithm 1 summarizes the first set of rules drafted by your company's developer. Considering these, answer the following questions.

	Column	Data type
1	rain level	float
2	river speed	float
3	atmospheric temperature	float
4	atmospheric pressure	float
5	wind speed	float
6	river level	float

	Column (cont.)	Data type
7	hour	int
8	minute	int
9	day	int
10	month	JAN – DEC
11	year	int
12	season	SPR, SUM, FAL, WIN

Table 1. Available data as input to the flash flood alert system.

```
# alert rules version 0.0.1
# Obs 1 and 2 dont agree on the nightly proc
# Obs 1 and 3 dont agree on the Summer proc
# Obs 2 has never seen a nightly alert

alert = “none”

if season = “SPR” {
  if hour > 18 or hour < 6 {
     if rain_level > 80 {
        if river_speed > 75 and temp > 25 {
           alert = “red”
        }

        else {
           alert = “orange”
        }
     }

     else if 40 < rain_level < 80 {
        alert = “orange”
     }
  }

  else if rain_level > 80 {
     alert = “yellow”
  }
}
else if season = “SUM” {
   if rain_level > 80 {
      if river_speed > 75 and temp > 25 {
         alert = “red”
      }

      else {
         alert = “orange”
      }
   }

   else if 40 < rain_level < 80 {
      alert = “orange”
   }

   else if rain_level > 35 {
      alert = “yellow”
   }
}

else {
   if river_speed > 75 and pressure > 1.0 {
      alert = “red”
   }
}
```

Algorithm 1. The first set of rules proposed by one of your developer peers.

[Question 1] (1 point)

Is the system proposed by your company’s developer and represented through Algorithm 1 an example of a machine learning (ML) solution? (1) Please justify your answer.

(2) If you think it is indeed an ML solution, (2.1) define what Task T, Experience E, and Performance P would be in such a case, and (2.2) explain the advantages of adopting ML.

(2) If not, (2.1) explain how you would make it ML and the advantages of doing so. (2.2) Define what Task T, Experience E, and Performance P would be in your new ML approach.

I do not think this is a machine learning solution. Machine learning is defined as something that gives the computer the ability to learn without explicit programming. In the above case, there is explicit steps that are taken to check what the alert level should be. In order to make it a ML solution, we can use logistic regression to classify what level the alert should be. This would help make it more accurate, more robust to new data, and help support strategy discovery. The task would be to figure out the alert level, the experience would be the 50 years of data collected, and the performance would be how well it can classify the alert level.

[Question 2] (1 point)
Congratulations! After discussing with your development team, you convinced them that adopting an ML-based solution to implement the system is indeed the best approach. The question now is whether the ML algorithm should provide as output (i) the alert severity level (namely *none*, *yellow*, *orange*, and *red*) or (ii) the predicted river level for the next hour or so. (1) From the standpoint of ML solution types (e.g., supervised, unsupervised, etc.), what are the similarities and differences between the two options? (2) What ML solutions (e.g., decision trees, neural networks, etc.) would you adopt for each situation? Please justify your answer.

I believe that the severity level would be a relatively
straightforward classification problem to tackle. There are
both supervised learning problems, for i, we have
a discrete output w/labels, so we have classification.
For ii, we have a continuous output, so we have a
regression problem. We could use either a decision tree or
recommendation system for i, where the severity levels are
the decisions/recommendations based off of given data.
For ii, we can use neural networks or linear regression
as the data is continuous so we can create a simple
regression model. We can say the similarities are the type
of ML problem they are, both supervised, and their difference
is that the output for i is discrete w/ labels and ii's output is continuous.

[Question 3] (1 point)
Another team member is worried about the data features that involve time, namely the *hour*, *minute*, *day*, *month*, *year*, and *season* items. She says some of these columns are even useless and should be removed. How would you pre-process each one of these elements to prepare the data for training and feeding ML solutions? Please consider any combination of the data pre-processing methods discussed in class and justify your answer.

I would pre-process the data by completing these steps:
- Converting the "month" column to a numeric (1-12) which makes it easier to feed into a ML algorithm
- Removing the "season" column because the "month" column will likely explain the same variation in the data and with more levels
- Removing the "minute" column because changes in flood risk likely don't change that frequently. The "hour" column is very likely sufficient to explain the time-of-day variation

We can use feature drop to remove some of the redundant
metrics. In this case we can drop year and season.
Year because weather changes over the years are consistent
and the year does not play a part. Season because we
can derive the season from the month and the day

After carefully analyzing each option, the team selected the approach of making the system provide the predicted river level for the next hour or so as output. To accomplish the task, the chief developer decided to adopt a third-party ML library that presents several *fit* and *predict* function pairs, one for each ML method. While looking for *predict* functions that output real values in the specs (to output the river level), they found the *linreg fit* and *predict* pair. Table 2 summarizes both functions.

linreg packet	
fit function **Input:** *(1) data, (2) degree, (3) sgd (yes or no), (4) lr (default=0.1)* **Output:** *model object*	*predict* function **Input:** *(1) data, (2) model object* **Output:** *real value*

Table 2. Input and output information of the *linreg fit* and *predict* function pair.

[Question 4] (1 point)
While testing with the *fit* function, the chief developer noticed it saves a couple of graphs that depend on the given values of *degree, sgd*, and *lr*. One of the graphs obtained is provided below.

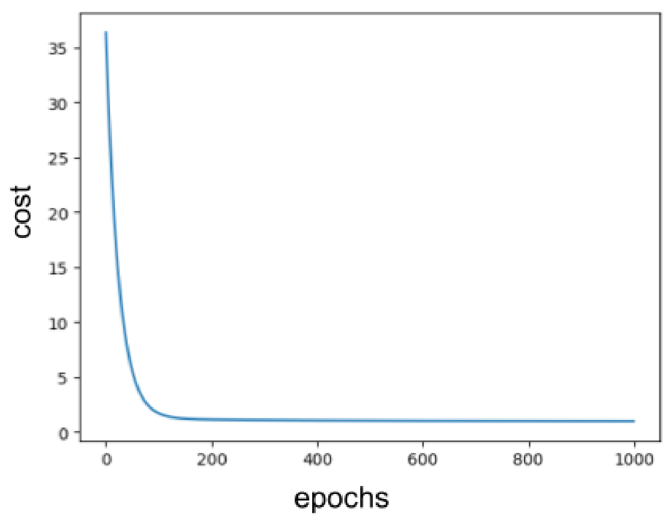

Figure 1. Cost record of fitting to the given data when *degree* is 1, *sgd* is *no*, and *lr* is 0.1.

The chief developer is intrigued by the decreasing behavior of the y-axis values. Based on your ML experience and the graph information, what is the meaning of the values expressed in the y-axis? How are they related to the concepts of *gradient descent* and *normal equation*?

The values that are expressed on the Y-axis is the cost function which is the mean-squared error in this case $= \frac{1}{n}\sum_{i=1}^{n}(y-y_i)^2$

This value determines how well our model is working. If we are having less cost value, that means we have less mean-square error, which means the model is performing well.

Both gradient descent & Normal equation are techniques used to reduce the cost function to its lowest. But Normal equation is an analytical approach while gradient descent is not. The equation for both gradient descent → $\theta_0' = \theta_0 - \alpha\theta_0$

Normal Equation → $\theta = (X^TX)^{-1} X^TY$.

[Question 5] (1 point)
After reading your explanation for the y-axis values, the chief developer is excited about the low values obtained after processing the entire data. Why should they be cautiously excited about it? What important ML protocol are they missing? What advice would you give the team to evaluate the generalization ability of the obtained ML models correctly? Please justify your answer.

The important ML protocol they are missing is to try it (the model) on the test data (unseen data). & also there is a problem of overfitting / underfitting. It is necessary to identify the validation set which constantly checks if the model is performing well or not. I would advise the team not to contaminate the data. I will ask them to first create training set, validation set & testing set. By evaluating the model on validation set & test set, then we will be able to confirm that the model is actually performing well.

[Question 6] (1 point)
Problem solved. The chief developer decided to follow your advice. Moving forward, while changing the *sgd* parameter to *yes*, they noticed a faster execution of the *fit* function, with some changes on the cost record graph. The new graph is provided below. In your opinion, what is the *sgd* parameter expressing? In what situations would you set it up to *yes*? Please justify your answer.

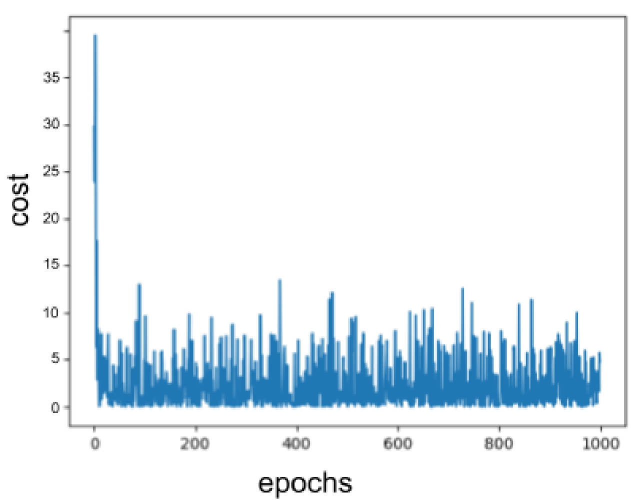

Figure 2. Cost record of fitting to the given data when *degree* is 1, *sgd* is *yes*, and *lr* is 0.1.

It seems sgd is indicating Stochastic Gradient descent, which uses fewer of the data values to test and is faster. Here I would set it to no since it doesn't seem too computationally expensive to use all the data, but if it took a long time, Stochastic GD is appropriate.

[Question 7] (1 point)
While changing the *degree* parameter, the chief developer noticed significant changes on another set of graphs, which are provided below.

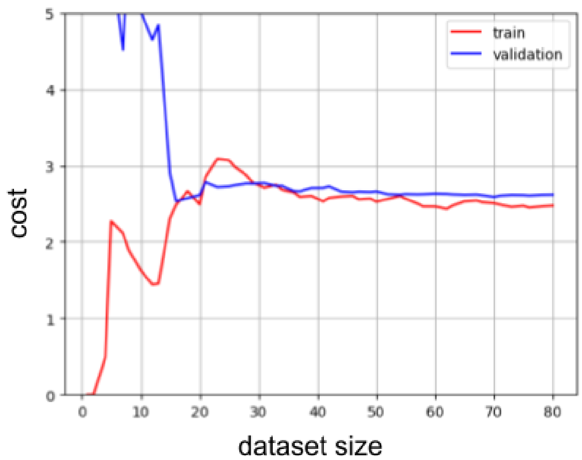

Figure 3. Test and validation cost records when *degree* is 1.

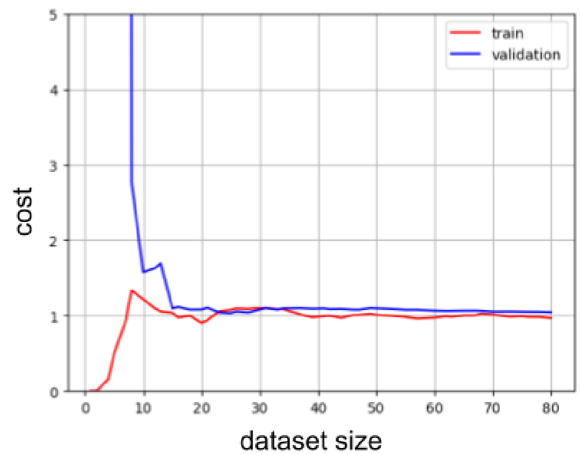

Figure 4. Test and validation cost records when *degree* is 2.

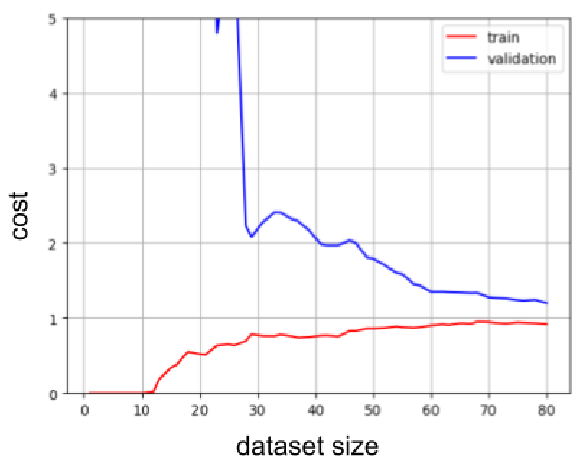

Figure 5. Test and validation cost records when *degree* is 10.

If you were to employ regularization, which of the three setups (figures 3, 4, and 5) would you apply it to? Please justify your answer.

Figure 5. By applying regularization, I would limit the degrees of freedom. Figure 5 is overfitting a lot in which regularization would help manage. You could try it on figure 4 but there would not be as much change. It would not work on 3 because it is underfitting.

[Question 8] (1 point)
Concerning the *lr* parameter, the chief developer noticed that setting it above 0.96 makes the *fit* function print a “no-convergence” warning message, with a final cost value that is surprisingly high (see the graph below). What should the meaning of the *lr* parameter be? When is it useful?

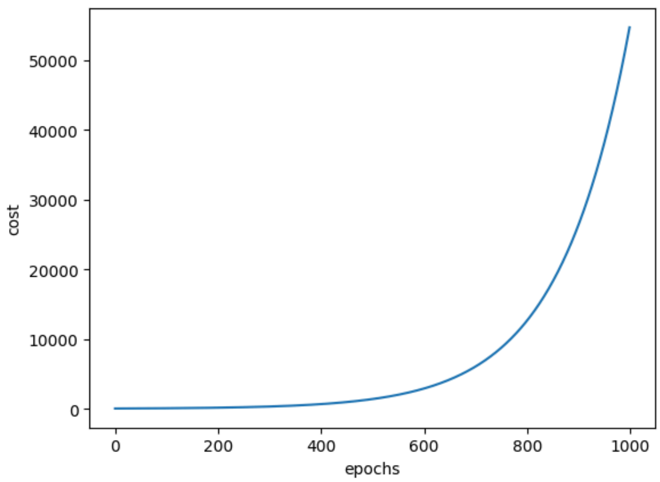

Figure 2. Cost record of fitting to the given data when *degree* is 1, *sgd* is *no*, and *lr* is 0.97.

lr = Learning Rate

This adjusts how quickly the model will change weights. Setting it too low may cause us to be inefficient or platue. Too high and we do not go in the right direction and dont converge. Useful for finding optimal training time.

[Question 9] (1 point)
What is PCA? What is it used for? What are its steps? How should one choose the value of *k* (i.e., the new data dimensionality after PCA projection)?

PCA is a standard practice that aims to tackle the curse of dimensionality by choosing to keep the most important dimensions that represents the most of the data's variance. ① Standardize the d-dimensional data ② Compute data covariance matrix ③ Compute eigen value and eigen vectors of the covariance matrix ④ select k ≤ D best eigenvalues and prepare the proj matrix ⑤ Project the d-dimensional data into k dimensions using W. you should pick k based off the amount of variance you want represented in your solution. 1.0

[Question 10] (1 point)
Please explain the no-free lunch theorem and its practical consequences to ML.

The no free lunch theorem means that there is no best model to use in ANY CASE. The consequences of this is that you have to test many models to see which one works best for your data. 1.0

No free lunch theorum is that if you don't make any assumptions about the options or solutions then there is no reason to chose one solution over the other. Therefore, an assumption must be made about the data in order to chose the best option.