Name: Jason Zhou
Mentor: Dr. Dongjin Song
Status Report #: 8
Time Spent on Research This Week: 6.25
Cumulative Time Spent on Research: 45.75
Miles Traveled to/from Mentor This Week: 0
Cumulative Miles Traveled to/from Mentor: 0
==========

Monday, November 1st, 2021: (0.25 Hours)

To begin the new week, I had a zoom meeting with my mentor. After telling me that I had been trying to learn linear algebra, he recommended a video course for me to watch. Although the course looks to be incredibly old (almost a decade), it is taught by an MIT professor and is more or less up-to-date (given that established math does not change). My mentor told me that it was not necessary for me to watch the entire series, as I really only needed to know up to the fifth lesson. Additionally, each online lecture looks to be around 45-60 minutes long.

Tuesday, November 2nd, 2021: (1 Hour)

As this day was election day, I had no school and decided to take this opportunity to watch the first video of the MIT linear algebra course my mentor had recommended to me the previous day.

I learned that in linear algebra, there are two types of representations: the row picture and the column picture. The row picture is simply the math that high schoolers use to solve systems. In contrast, the column picture uses matrices to form a single equation instead of a system of equations. It was here that the professor explained that this is what is called a linear combination. Essentially, it is a combination of values for certain variables that allow one to get a certain output.

Additionally, I learned of the Ax = b formula where A is a matrix of coefficients, x is a vector of variables, and b is the output. Again, the linear combination concept applies. If by multiplying matrix A by vector x (which can hold any linear combination), one can output any solution, then the equation is considered to be invertible. However, if the equation only gives an output for certain linear combinations, then the equation is said to not be invertible. As of right now, I am unsure why an equation being invertible or not would be significant, so I will have to look further into this at a later time.

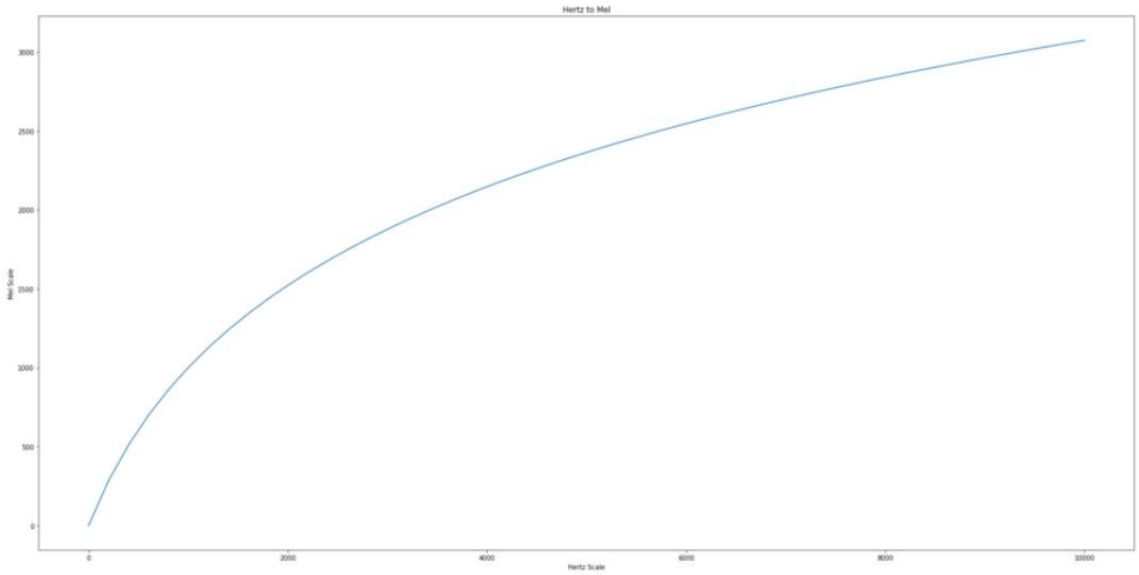
Thursday, November 4th, 2021: (1 Hour)

On this day, Ms. Pintavalle let my class have in-class time to work on our research proposals. As such, I worked on writing the introduction of my research. Mostly working off of what I had written in the Step 1 document for gathering info, I started with a broad explanation

of what my topic was and how it could be used in the real world. Then, I narrowed down the topic by explaining the different approaches that had been attempted. In terms of the “And, But, Therefore” formula, I have only written the “And” part, so I will have to work on the “But” and “Therefore.”

Friday, November 5th, 2021: (1 Hour)

On this day, I began researching what a mel-spectrogram was. From background knowledge and what my mentor briefly explained to me, a mel-spectrogram is the visualization of sound in the form of a graph. After looking at some articles on the internet, I found that, as the name implies, a mel-spectrogram is a combination of the use of a spectrogram and mel-scale. A spectrogram is just a way to visualize the data while the mel-scale is a frequency scale that is used to better model the human perception of sound. For example, humans can tell the difference between 100 and 200 Hz; however, they cannot tell the difference between 1000 and 1100 Hz, despite their being the same gap in frequency. This is because the higher the frequency, the harder it becomes for humans to discern between sounds. Thus, the mel-scale attempts to model this through a log curve.



(An example of the mel-scale graphed as a log function. The x-axis starts at 0 and is incremented by 2000 every tick. The y axis begins at 0 and is incremented by 500 every tick.)

Take the graph above as an example of the mel-scale. The distance between low frequencies has a large distance between them on the y-axis, whereas the higher frequencies have smaller distances between them on the y-axis. The general idea is that frequencies that are equal distances away on the mel-scale also model what humans would perceive as equal.

Saturday, November 6th, 2021: (2 Hours)

On this day, I decided to watch the second video in the MIT linear algebra series that my mentor recommended to me. The topic for this video was on the forward elimination of matrices, which like regular elimination in algebra, is a way to calculate for unknown variables within a system. However, this method is more streamlined and easier to do when working with more complex systems. For example, if we have the system of equations:

$$\begin{aligned} x - y - z + u &= 0 \\ -y - 2z &= 8 \\ 2x + 2z &= -8 \\ 3x - 3y - 2z + 4u &= 7 \end{aligned}$$

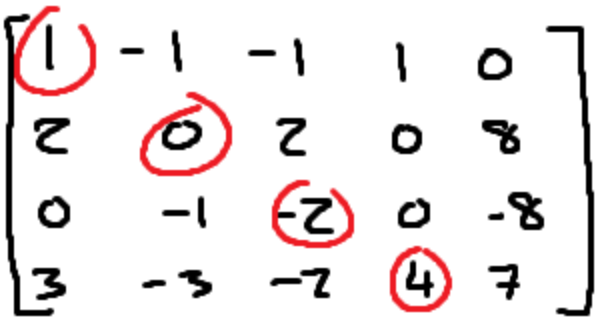
(This is a picture of a system of equations with 4 unknown variables: x, y, z, and u)

Then the following matrix can be formed by taking the coefficients of each term and output of each equation:

$$\begin{bmatrix} 1 & -1 & -1 & 1 & 0 \\ 2 & 0 & 2 & 0 & 8 \\ 0 & -1 & -2 & 0 & -8 \\ 3 & -3 & -2 & 4 & 7 \end{bmatrix}$$

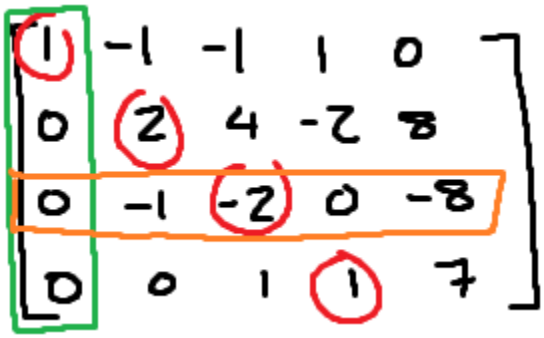
(A picture of a matrix that was converted from the system of equations above. The rightmost column is the one with the answers to each equation in the system. Also note that the leftmost column to the rightmost column corresponds to x, y, z, and u, respectively)

After the matrix has been formed, one would identify, which always starts at the top left and goes one space down and one space to the right (diagonal) until the last row is reached.



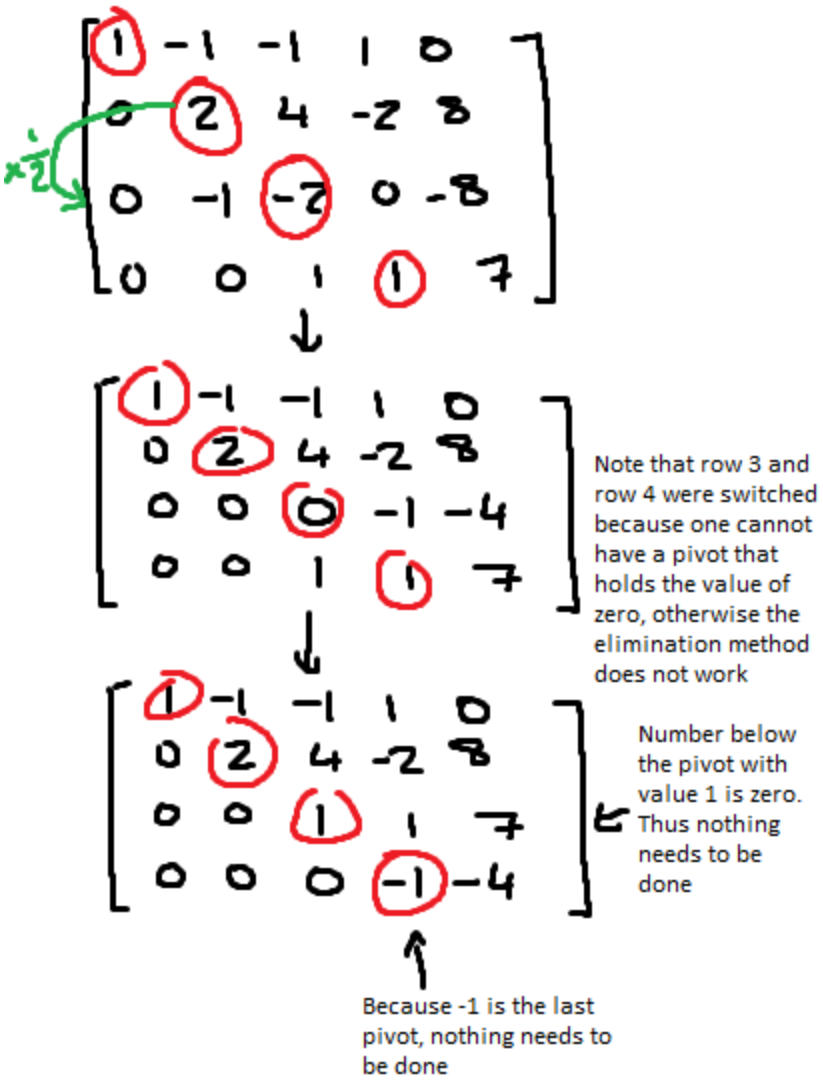
(A picture with the same matrix but with the pivot points identified)

Starting at the leftmost pivot point, one would make sure that all numbers below it equal zero. This is done by multiplying the pivot point by a certain value so that it adds or subtracts to the bottom number to equal zero. So, in this case, one would be multiplied by -2 to cancel the 2 right below it and would also need to be multiplied by -3 to cancel the 3 below it. Keep in mind that when multiplying the pivot will also multiply the rest of the numbers within the row and cause those numbers to be added/subtracted to the numbers below them. It is also worth noting that the 3rd row does not need to have anything done to it because the first number of the third row is 0. Thus, after the process is done, the matrix looks like this:



(A picture of the matrix after the first row was multiplied by certain numbers to zero out the first column. Although the first column is used to multiply and zero other rows, it does not change within the matrix. Notice in the column marked in green, all values below the pivot point became zero. Additionally, notice that the row marked in orange remained unchanged because the value below the pivot point was already zero, therefore nothing needed to be done)

After this is done for one pivot point, then it must be done for the next. This process is repeated until one reaches the last pivot. During the last pivot, however, nothing is done and the elimination method is concluded. Starting with the above matrix and continuing with the above example, the rest of the calculations would look like this:



(A picture that displays the rest of the forward elimination process for matrices)

Now that elimination has been performed. The last matrix in the above image is the final matrix that holds the answers that solve the system. To actually translate this back into variables that one can recognize, a process called back substitution is used. Remember that each column represents a variable such as x,y, z, and u respectively, with the exception of the rightmost column (because it holds the actual outputs of each equation). Knowing this, one can take the last row and translate it back into an equation. -1 indicates the coefficient for the 4th column, which would be u, while -4 represents the output. Thus, the equation -u = -4 can be written, which simplifies to u = 4. The equation above the last row would come out to be z + u = 7 (because the coefficients in the appropriate columns are one, and the output column is 7). Because “u” is known, the value can be substituted to look like this: z + 4 = 7, which simplifies to z = 3. This can be repeated until one reaches the first row, which is where all variables can be calculated for. This is why it is called back substitution. One starts from the last row and ends with the first row. Doing out the math, x = 1, y = 2, z = 3, and u = 4.

Sunday, November 7th, 2021 (1 Hour)

For the final day of the week, I wanted to try experimenting with an idea I got for detecting anomalous sounds within an environment. Instead of working with sounds, however, I used a common image dataset in TensorFlow which is well known within the machine learning community. The reason was that it was just easier to access and load up into my coding environment.

Autoencoders do is take an input, compress it, and then decompress it through reconstruction of the data. The intent, thus, is to give the model an input and expect that it returns the input as an output (it will not be identical but similar because some data is lost in the process). One of the inherent properties of autoencoders is that they tend to only perform well when asked to classify data that was similar to their training data. If they are given data that is dissimilar, the reconstruction of the input will be poor. As a result, autoencoders have become popular in anomalous sound detection systems. One can simply train an autoencoder on normal sound data, and if an irregular sound is inputted, then the reconstruction will be poor. By doing this, it becomes easy to identify the data with irregular sounds (because they will be reconstructed poorly). However, the difficulty lies with audio data with irregularities that sound *normal*. Simply put, it is hard to discern whether the audio is normal or abnormal. If the abnormal data appears normal, although the data might be reconstructed worse than normal audio data, it will be hard to definitively categorize it as an anomaly.

As such, I wanted to try to use two autoencoders instead of one to solve this problem. To explain, the first autoencoder will reconstruct the input (we will call this i). Then, this reconstructed output (o) will be inputted into the second autoencoder (we will call the input i2). By doing this, the reconstructed version of i2 will be made (this will be referred to as o2). Because o2 is a reconstruction of a reconstruction, o2 will have lost more data than the original i. In theory, if this process is performed with normal audio data and irregular audio data, then the irregular audio data will have a more pronounced reconstruction between i and o2. If one applies this to irregular audio data that sounds normal, then the differences between that and normal data will be easier to see. Ultimately, this would allow the model to more easily categorize anomalies as anomalies. Of course, this process does not have to be limited to just two autoencoders, but as many autoencoders as needed (one can even re-use one autoencoder and just keep feeding it the reconstructed output as an input).

I went into my coding app and decided to test whether this would really work. Specifically, as a starting point, I just wanted to know if the model truly reconstructed the input worse and worse each time. Here is the result of my simple experiment:

tf.estimator.DNNClass | Kernel method - Wikip | Feature (machine learn | ARM B 2021-22 | trying to take an outp

colab.research.google.com/drive/1Y7vfJG0GG4Qfzn-0JUwkps-WNRBQxdiZ#scrollTo=gtg9l0NfY88i

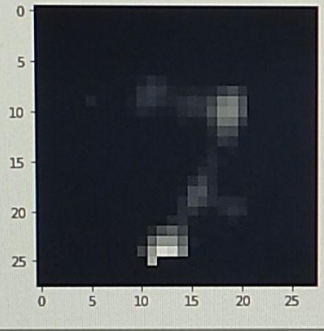
Autoencoder

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

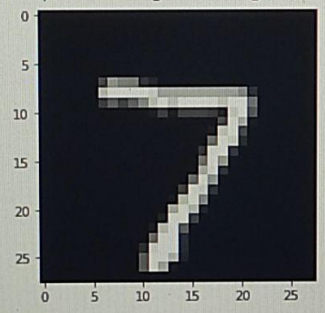
```
input = ae_out

for i in range(10):
  test2 = autoencoder2.predict(input.reshape(-1,28,28,1))
  test2 = test2.reshape(28,28)
  input = test2

plt.imshow(test2, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7fa57404e790>
```




```
[22] plt.imshow(x_test[0], cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7fa576d9d410>
```



0s comp

Type here to search

(An image of my experiment. The top image is the reconstructed image of a seven after it was put through 10 autoencoders. The bottom image is the original input that was put into the first autoencoder. I have left out the code for the autoencoder to keep the meaning of the image concise and less confusing)

As one can see, the output was reconstructed worse and worse. Ultimately, making it barely visible in the final reconstruction (with each reconstruction, the image lost more and more data, resulting in a less visible seven). Ultimately, I would like to see how I can apply this to sounds and am excited to see what the outcome will be!

References

Gartzman, D. (2019, August 19). *Getting to know the mel spectrogram*. Towards Data Science. Retrieved November 8, 2021, from https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0

mlearnere. (n.d.). *Learning from audio: Spectrograms*. Towards Data Science. Retrieved November 8, 2021, from https://towardsdatascience.com/learning-from-audio-spectrograms-37df29dba98c

mlearnere. (n.d.). *Learning from audio: the mel scale, mel spectrograms, and mel frequency cepstral coefficients*. Towards Data Science. Retrieved November 8, 2021, from https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8

mlearnere. (2020, September 16). *Learning from audio: wave forms*. Towards Data Science. Retrieved November 8, 2021, from https://towardsdatascience.com/learning-from-audio-wave-forms-46fc6f87e016

Strang, G. (n.d.). *Lecture 1: The geometry of linear equations* [Lecture transcript]. MIT Open Courseware. Retrieved November 8, 2021, from https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/lecture-1-the-geometry-of-linear-equations/

Strang, G. (n.d.). *Lecture 2: Elimination with matrices* [Lecture transcript]. MIT Open Courseware. Retrieved November 8, 2021, from https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/lecture-2-elimination-with-matrices/