Jerry Chen

CS 541

Artificial Intelligence

Prof. Shen

HW2

For this homework, I just want to kind of review some of the topics that we have covered in the past lectures and write down my understanding on them. I have taken quite a few math classes in the previous and current semesters on calculus, probability and linear algebra but it isn't really easy for me to follow the professor's pace during lectures since I kind of get lost in all of the proofs for different formulas. Therefore, for a lot of the topics I have went online to various math/tech websites to get a better understanding of how they work and try to relate them back to the course material. In particular I find that the pure theoretical topics like the JL Lemma and logistic regression to be harder to understand since its difficult for me to actually visualize it in my head while the actual algorithms like random projection and PCA are easier to visualize and grasp. Hopefully going over these topics that we have covered in class so far would also help me refresh my memory to prepare for the midterm coming up.

- Markov's Inequality: $P(X \geq t) \leq \frac{E(X)}{t}$

Works around with two values, the expected value of X, $E(x)$, and the independent variable t which we want to test. The inequality would return the probability in which a random variable X would be greater or equal to the value t and we have chosen. As the wording of the inequality suggests, the probability that X is greater than or equal to t is less than or equal to the expected value of X over t, the greater we set our variable t to the lower the probability would be, and if t is set to be $E(x)$, then the probability would be 100%.

- Chebyshev's Inequality: $P(|X - E(X)| \geq t) \leq \frac{Var(X)}{t^2}$

Similar to Markov's inequality, but slightly more complex as it introduces another variable variance of X. The inequality states that the probability in which a random variable X is greater than or equal to the value t away from its expected value is less than or equal to its variance over t squared. It requires us to know information about the variance in addition to the expected value but is stronger than Markov's since it tells us the probability on a range of possible X's. We can also use it to find the confidence interval of X by setting the right-hand side to a confidence percentage, say 95%, and then use the inequality to find the interval.

- Random Projection

The purpose of random projection is to reduce dimensionality in high dimension data and to represent them in a lower dimension space without losing out too much of the original information or data. The basic idea behind it is that if we have data that is represented by a n*d matrix x where d is a very high value, we can reduce that dimension down to a smaller value k by creating a random projection matrix A of dimension d*k with points being randomly generated by Gaussian distribution with $\sigma^2 = \frac{1}{\sqrt{k}}$ so that the Euclidean distance between points would be preserved in the reduced matrix A*x that would only

have the dimension n*k. Now we are left with a reduced matrix projected from x that is much smaller to represent yet still mostly retains the Euclidean distance between points of x due to the Gaussian distribution.

- Johnson-Lindenstrauss Lemma

The JL Lemma from my understanding is not an algorithm but rather a proof saying that if we have some data with N points each represented by N coordinates, then we are able to map that data to a smaller logarithmic space with N points represented by log(n) coordinates instead. The Euclidean distance of the points will be preserved by the factor of $(1 + \varepsilon)$, but there would be some small error being introduced to the data in exchange for this dimension reduction. Moreover, the lemma could be applied multiple times to improve the probability of getting a 'perfect' reduction, so you could just set the error or confidence percentage that you are willing to take and the lemma will give you the space that captures distance up to that error percentage.

- Principle Component Analysis

PCA is another dimensionality reduction method similar to random projection but in a more systematic and less random way that will only produce one possible result at any run rather than the random result that we would get from random projection. The way I understand how PCA works is that given a dataset with N dimensions, we construct the covariance matrix of the data and then compute the eigenvectors and eigenvalues of the matrix. The number of eigenvectors would match the number of dimensions, so for 3 dimension there would be 3 eigenvectors. For each eigenvector represents a direction and is paired with an eigenvalue, the higher the eigenvalue the more information its corresponding eigenvector carries. So with each eigenvector being a principle component of the dataset, we rank them based on their eigenvalue and because the higher-value principle components carry more information, by taking the highest ranking principal components and dropping the lower ones we can create a new dataset with less dimensions but still retaining most of the information since the low ranking principle components hold miniscule amounts of data compared to the high ranking principal components. By deciding how many principle components to cast out we can decide how small and accurate the final result will be to the original dataset.

- Singular Value Decomposition

To my understanding SVD is very similar to PCA in that it also decomposes matrixes into separate components, for SVD it breaks up a matrix into three separate matrixes $A = USV^T$ where S is a diagonal matrix with all values outside of the diagonal being 0. By taking only the largest single values of S and setting the other to 0, we could reduce the rank of the data to however many values we keep and because the largest single values hold the most information, discarding the rest won't affect the accuracy of the resulting data by too much. In this aspect it is very similar to PCA with how the lower rank principal components are discarded. With SVD, it also means that we do not have to remember the whole matrixes of U and V^T since many columns would get dropped with the zeroing of singular values on the diagonal matrix.

- Recommender System

We start with a sparse observed matrix that records the inputs from users on a multitude of different items. The input could be the user's ratings on the items that they have purchased before and since its

rare that anyone has bought all of the items, the observed user-item matrix would be quite sparse. The job of the recommender system would be to find users that have similar ratings on certain items that they have both purchased and use that connection to predict how one user would rate another item that they personally may not have bought before, but someone similar to them has. The system would take the other user's rating to fill in the empty cell and vice versa to fill out the observed matrix. The fully filled out matrix would be a true preference matrix with a low rank since some of the rows would be the same after the recommender system predictions. With collaborative filtering we can decompose a true preference matrix into two separate matrixes, an user matrix and an item matrix which we can use instead of the original observed matrix to save computational time. Now to make a recommendation we can just multiply the user vector to the item vector to get an estimated rating which we will use to decide whether or not to recommend that item to this user.

- Binary Feedback

We talked about binary feedback where the values in our matrix would be converted using a sign function to output a matrix with either -1 or 1 to indicate whether or not a user would like or not like an item. This saves storage and is an easy way of representing our data but destroys low rank structure and is impossible to recover information about the original matrix, meaning we can no longer predict the user's true preference. The solution was to use a logistic function instead of the sign function since the logistic function sets the probability that a user rating returns 1 for an item to be higher according to their true preference rating and therefore the better rating users gave an item, the more positive 1's we would get and be able to use that to recover true preference.

- Logistic Regression

This is probably the topic that I had the most difficulty understanding and following during lecture since it was tying in maximum likelihood estimator and a lot of other math concepts, but from what I am grasping it is model used to predict the probability of a certain outcome. Unlike linear regression it is not really for continuous outputs, but rather definitive events such as 'yes' or 'no'. For our example of getting a binary feedback from the recommender system our two possible outcomes will be -1 or 1 representing like or unlike. So, the higher our input (true preference) is, the more likely it will be for the event of 1 (like) to happen and the lower our input the more likely our output would be -1 (dislike). This would still be a reliable way to gauge whether or not a user would like an item on a large scale and we would still be able to recover the true preference that the user has on an item by simply running the function multiple times to see the probability.

So that should be most of the topics that we went over in lecture for the past couple of weeks, I feel like I have a good understanding on most of the material from the earlier lectures and with how dimension reduction works in general. The more recent maximum likelihood estimator and logistic regression are still quite confusing and hard to follow, but hopefully they will become clearer as I get more time to digest the information. I am finding out through this course and Computer Vision that linear algebra really is important in representing data in computers and that there are really a lot of different methods to manipulate data matrixes when approaching all these different types of problems. I honestly should have taken linear algebra before taking this course and CV, but unfortunately, I am actually taking it this semester as well so it's sort of a learn as I go situation with the occasional skipping ahead. Nevertheless, I feel like I am learning a lot in this course about the workings of machine learning/AI.