**CSE514A Datamining – Fall 2019 Course Project Note: this description will be updated as we go**

**Version on October 10, 2019**

**Preamble:**

1. This course project has multiple parts, and will be done progressively in sequence. This means that a late part will depend on the previous parts; therefore, it is important to make sure you get each part right before proceed to the next. Each part will also be graded and counted toward the final project score.
2. The project will start with some detailed description on what need to be done, which is more in line with a homework assignment, and progress to some open-ended questions on models/patterns to be learned and the features/properties of the expected models/patterns.
3. The project involves reading some online materials and papers, programming, discussing ideas among team members if needed, and deep analysis of the data used and anticipated results.

The overall project is built around Variational Autoencoder (VAE) and serves multiple purposes.

**The objectives of the first part of the project:**

1. Generate some images that are similar to, but not the same as, a set of initial given images. Here, we hope to develop a new image reconstruction model based on some training data.
2. Understand how VAE is able to extract and learn image specific features in a latent, low-dimensional space.
3. Understand how model complexity affect the quality of reconstructed images.
4. Learn and understand a *component representation* using the VAE model.

**What to do – Part 1: MNIST images on hand written Arabic numbers 0~9**

* Step 1.1 – Set up your working/programming environment for the project and choose the programming language you are good at. Since deep learning is popular and becomes dominant, I highly recommend to use python, which is almost the default language for deep learning. ***In addition, having access to GPU will be mandatory for the project since you otherwise won’t be able to accomplish what are required.***

*Recommendation*: I highly recommend Google's Colab where Jupyter notebooks and other related tools are available so you can run your programs on Google's GPU machines for free if you don't have your own GPU box.

*Disclaimer*: Since this is not a programming course and programming is prerequisite for the course, we may not provide any assistance to your programming issues associated with this project.

* Step 1.2 – Let’s start with “Visualizing MNIST with a deep Variational Autoencoder” (<https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder>). Read it closely and try your best to understand the mechanism used (which should in principle follow VAE we discuss in class). Make sure you complete this step with no running error before proceed further.
* Step 1.3 –Now, we are going to study the effect of the dimension of the latent space. Figure out where the latent variables are specified in the program you use.
  + Step 1.3.1: Change the dimension of the latent variables from 10, 8, 6 until 2 and 1. Let’s name the VAE model with k dimension (k=10, 8, …, 4, 2, 1) specifically M(k). For each dimension, plot and visually inspect some images from the output layer to get some rough idea of the image quality.
  + Step 1.3.2: Search on the web to find one or two reasonable methods for comparing the similarity of two images. Here we want to directly compare two images in the input (or original space).
  + Step 1.3.3: Now we can use M(k) to reconstruct a given input image. Let’s call an input image im(\*), and name a reconstructed image from M(k) as im(k). Using the similarity measure from Step 1.3.2, compute the similarity between im(\*) and im(k). Plot the average similarity overall 100 input images as k changes.
* Step 1.4 –Now, we are going to study the effect of the number of layers. Figure out where the latent variables are specified. Note that the code for Step 1.2 uses two hidden layers. (note: this may not be true, so double check)
  + Vary the number of layers from 1, 2, 3 and 4 layers and compare how the number of hidden layers help increase or reduce image quality. Here you may fix the dimension of the latent variables that give you the best image quality.

**What to submit for Part I** – it’s ***due on Friday, October 25***. You submit on canvas – instruction to come later

* You must submit the first part of your project report, ***independently and exclusively done by yourself*** – see collaboration policy in the course administration document which is available on course canvas.
* Submit (1) an *MS Word file* *for your report* (so I can give you feedback in your file) and (2) *your program or package of programs*. **I may ask you to demonstrate how you generate some of your results**.
* What to include in your report – total 10 points.
  + Anything interesting that you like to discuss for Steps 1.1 and 1.2. This is more for yourself for bookkeeping what you did differently from and/or beyond the instruction. (1 point)
  + Step 1.3.1: Discuss where in your program the variables are for the latent variables of your model. Discuss how the dimension of the latent variables affect the quality of the output images. (Tip: you may want to use a plot to show your results.) Included in your report some images you generated – if possible, let your images tell a story behind what you observe. (1 point)
  + Step 1.3.2: Discuss what similarity function you used to compare image similarities. Ideally, you may want to consider more than one similarity function – you will receive some bonus points if you consider more than one. (1 point, 1.5 if you use two)
  + Step 1.3.3: Plot figures as specified using your similarity function(s). (2 points)
  + Step 1.4: This should be self-explanatory as specified. Also plot your results. (4 points)
  + What you learned from this exercise – Discuss what you learned by the model(s) of the MNIST images that you created – this is an open-ended question, so think deeply regarding what you have been doing and be creative. Example questions include 1) why can a low dimensional latent variable or variables represent fairly accurately the diverse input images? 2) what image features do the latent variables capture? (1 point)

**The objectives of the second part of the project:**

Note: Part I is just for warming up, Part II is getting interesting…

1. Further understand how different image components are mapped to different components in the latent model.
2. Generate images that have *prescribed “distances” to or similarity with* the given set of images.

**What to do – Part II: MNIST images – components and controlled similarities**

* Step 2.1.1 – *Rationale*: Following Part I. We can learn a model, i.e., a latent vector variable, of a set of images. Let the latent variable for a set of MNIST digits 0~9 be V0~9. Likewise, we can learn a model V1~9 for MNIST digits 1~9 (without images for 0). Then, the difference between the two models V0~9 and V1~9, i.e., V0’ =V0~9 –V1~9 which is the difference between two vectors, should be a model for MNISST digit 0. (Note that we need to compute the difference for the mean vector and standard deviation.) We can also directly learn a model V0 for MNIST digit 0 using images for 0.
* Step 2.1.2 – Using the (best) method you developed in Part I, learn Vk and Vk’, for k=0, 1,…,9, as sketched above, using all the images in the MNIST dataset that you can use. Compare Vk and Vk’: Now, consider to use two dimensional hidden variables and plot 3-D normal distributions showing both Vk and Vk’ and compare. Also compare the quality of the images from the two models for each k.
* Step 2.1.3 – We can use the decoder for Vk to reconstruct images for digit k. Can we somehow reconstruct images of digit k using the model Vk’ ? The real challenge is that we don’t directly have a decoder for Vk’.
* Step 2.2.1 – *Rationale*: VAE is good for generating images that are similar with but not exactly the same as the original images. But the reconstructed images still follow the same distribution of the input images. How can we generate images that follow a similar but not exactly the same distribution of the input images? We want to do so by creating a new model N’ that is a variant of the learned model N. We can shift the mean vector or the standard deviation vector of the learned model.
* Step 2.2.2 – Based on the work in Part I and Step 2.1, use VAE with a hidden model of the minimal dimension that is able to reconstruct *decent* images. (I bet you would be able to use 2 or 3 dimensions.) Learn Vk, for k=0, 1,…,9. Using a small quantity δ, e.g., δ = 0.05, add i×δ, for i = 1, 2, 3, …, 20, to each dimension of the mean vector, and then reconstruct images using the original decoder of the corresponding VAE. Repeat this by adding i×δ to each dimension of the standard deviation vector. Generate 20 or more images for each value of i, and visually inspect the output images.
* Step 2.2.3 – Compute the average similarity between 100 (or 500) pairs of input and output images using the similarity function you used in Part I.
* Step 2.2.4 – A possible issue to consider: We will use the original decoder that was learned from learning the model Vk, but we will use a revised model (with mean or standard deviation changed). Will the decoder be consistent with the revised model? Yes or no and why?
* Step 2.2.5 – Optional: what if we consider the model using images for digits 0~9 in all questions in Steps 2.2?

**What to submit for Part II** – it’s ***due on Wednesday, November 6.*** Similar to Part I, you submit on canvas

* You must submit the second part of your project report, ***independently and exclusively done by yourself*** – see collaboration policy in the course administration document which is available on course canvas.
* Submit (1) an *MS Word file* *for your report* (so I can give you feedback in your file) and (2) *your program or package of programs*. **I may ask you to demonstrate how you generate some of your results**.
* What to include in your report – total 10 points.
  + Step 2.1.2: Report Vk and Vk’, for k=0, 1,…,9. Plot 3-D normal distribution images showing both Vk and Vk’ for the case when your model has 2 dimensions. Also include 10 reconstructed images from Vk, for each k=0, 1,…,9. Compare the quality of the images from the two models for each k. Plot your results. (3 pts)
  + Step 2.1.3: Discuss any idea that you can come up with for generating digit k using the model Vk’. (1 pt)
  + Step 2.2.2: Report the vectors of Vk, for k=0, 1,…,9. Report your images for different variants. (2 pt)
  + Step 2.2.3: Plot the similarities for each Vk, for k=0, 1,…,9 and its variants. (3 pt)
  + Step 2.2.4: A (possibly deep) discussion and analysis of this issue. Note that this is open-end question so be creative based on your understanding of VAE and your experience doing this project. (1 pt)
  + You may report your results using all images for digits 0~9 (instead of individual digits). Follow the same instruction above. (bonus 2 pts)

**The objectives of the third part of the project:**

* To work on datamining problems that you are interested.

**What to do – Part III:**

***Part III a -***

You are welcome to propose your own project. I you propose your own project, you need to submit a short proposal (~2 page write-up) and have my approval before you proceed. You may form a team up to 3 students (no more) for your project.

Your proposal must include information of

1. the problem(s) you want to solve/analyze,
2. the data you have in hand,
3. the methods you are going to use – you must consider at least two methods for comparison purpose.
4. the objectives you aim to achieve.
5. Team members and partition of the work, if any.

The deadline for proposing and submitting your proposal is Friday, Nov 8.

Submit a ~5 page progress report (single column, single space, 11 pt fonts, 1 inch margin) on Friday, Nov 15. By this point, you should have implemented your methods.

Submit your final report (at least 10 pages) on Friday, Dec. 6. The port must have all the components listed above. A template report will be provided later.

***Part III b -***

You may also work on a project I specify here. This will also serve as an example for your proposal if you choose to work on your own.

If you choose this project, you don’t have to submit a proposal, but you need to post your team and teammembers on piazza.

***The problem and objectives***:

I mentioned that my group has been working on a new class of RNA, called interior circular RNA or i-circRNA, and we have published our first paper (title: Interior circular RNA, RNA Biology, 2019, a copy of the paper is provided on the course canvas, along with the other papers mentioned below. Also see a news on this - <https://engineering.wustl.edu/news/Pages/Big-data-analysis-leads-to-discovery-of-new-class-of-RNA.aspx>). The work in our first paper is mining of a large quantities of gene expression data, plus biological experiments.

The main problem we want to address is to predict circular RNA (also i-circRNA) without gene expression data at all, but instead using genomic sequences. A related problem is identification of sequence features that are involved in the biogenesis (i.e., production) of circRNA. So our final products (or objectives) are a new method and software for *de novo* prediction of circRNAs using genomic sequences and a set of features of circRNAs.

***The data to use***:

Two sets of sequence data will be provided (coming soon), one for positive cases and the other for negative cases for training a classification model.

***The methods to use***:

We will consider to use Convolution Neural Networks (CNNs). We want to develop a CNN model for prediction of circRNA. As no one has done it before, we will try to borrow some ideas people have used for a similar problem called prediction of alternative splicing. For this latter problem and the methods proposed, I upload the following papers that ***you should read closely*** –

* *The splicing code goes deep* – this is a mini review of the work on splicing, focusing more on the recent work. This is for background reading for the problem of splicing.
* *Predicting splicing from primary sequence with deep learning* – this is one of the papers previewed in the previous article. It uses CNN.
* *Deep-learning augmented RNA-seq analysis of transcript splicing* – the problem considered is the same, but the data used are different. Here it uses gene expression data (i.s., RNA-seq data) rather than genomic sequence. But we can also get some idea what CNN architecture is used here.
* *Deep splicing code: classifying alternative splicing events using deep learning* – this focuses on classification using deep learning.
* *Deep learning for alternative splicing* – this is a nice course project report. We can get some idea how other course project worked on similar problems. More importantly, it provides some idea of CNN for the problem.

***What to do and schedule***:

What we need to work on, after reading these papers closely, is to develop a CNN based method for predicting circRNA from genomic sequences.

To-do list and due date:

* Due on Wednesday, Nov 6. Form a team of up to 3 members, report your team on piazza
* Due on Friday, Nov 15. Submit a ~5 page tech report (single column, single space, 11 pt fonts, 1 inch margin) to summarize what you learn after reading those papers.
* Due on Friday, Dec. 6. Submit your final project report (with the ~5 page tech report included).
  + Design your method for circRNA prediction. Discuss the CNN architecture and each individual component, test results and performance measure.
  + Design a method to identify the sequence features associated with circRNA bac-fusion points. Discuss how your method work and what features you discovered.

***Team members***: You may have up to 3 team members for this project. You should also count me in as a virtual team member and your cheer leader.