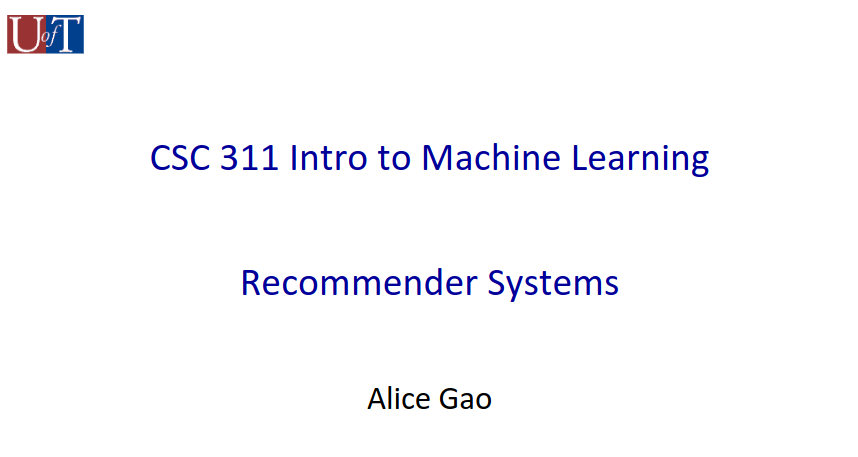
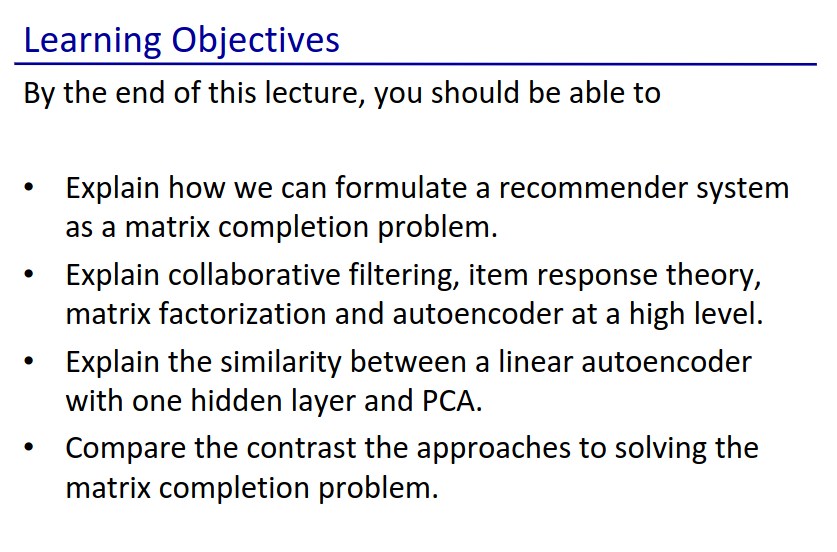
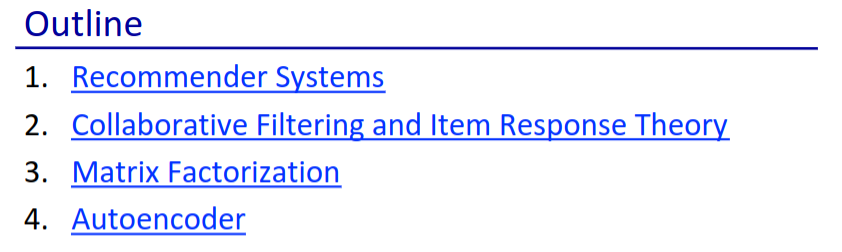
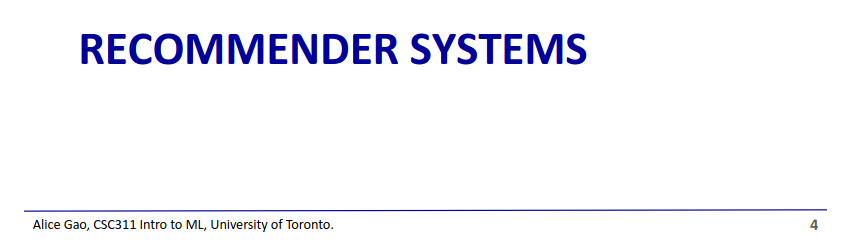
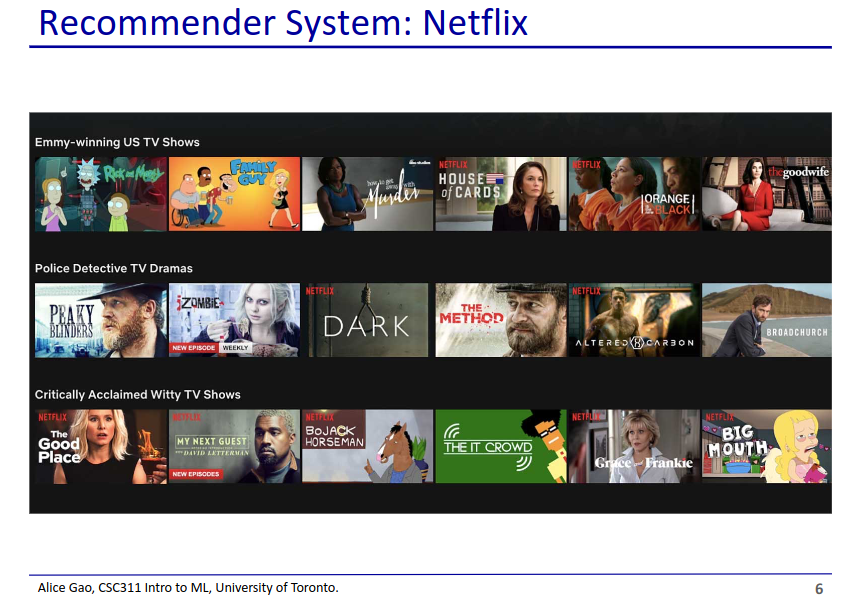
| **Admin**   * Last lab is this week - Naive Bayes   + Lowest lab is dropped, but please don’t skip this lab it is not hard * Test 2 is marked, will be released later   + Median is 75% * Final   + Questions that might appear in the final that may be harder than the midterms     - Derivation problems     - Conceptual problems   + Review session will be available   + There will be no practice problems   **Recommender systems**   * Models that try to provide recommendations of items that it predicts will score highly   + Algorithm that tries to predict which shows a user will want to watch on Netflix * We can represent this problem as a matrix completion problem   + Give the model an incomplete matrix R and have it fill in the missing spaces * **Problem for class assignment**   + Predict whether a student will correctly answer a question they have never seen before   **Collaborative filtering with KNN**   * Using KNN to fill out the empty spaces based on the most similar rows/columns of the matrix * **User-based approach**   + We look at which users (students) have similar performance to the user we want to predict for, and then assign the missing classes according to the k most similar users * **Item-based approach**   + We look at which items (questions) were answered similarly to the item we want to predict for, then assign the missing classes according to the k most similar items * Example on slide 13   **Item response theory**   * We define some latent variables that help us to define the probability that the missing entry is a given class   + Latent variables are variables stored by the model, but that we don’t observe * **In the case of our assignment problem:**   + Latent variables: each student has an ability score and each problem has a difficulty score   + The probability that student i answers question j correctly:   + We learn the latent variables using maximum likelihood     - Find the log-likelihood     - Find the derivative of the log-likelihood with respect to and     - Equate the derivatives to 0 and find and   **Matrix factorisation**   * Matrix factorisation is actually a dimensionality reduction model, but can be used to solve this problem * We represent each user and item as a vector with K latent factors   + The dot product of the vectors is the prediction to put in the matrix   + K is a hyperparameter * When fully vectorised::   + We have N users: **U** has dimensions   + We have M items: **Z** has dimensions   + We want to find N and M such that     - Not all values of R are filled in, so we just need to make sure the existing ones match * Solving for **U** and **Z**   + We try to optimise     - O is the observed (existing) entries in R (we only optimise with respect to the entries in R that we know)   + This problem is very hard to solve if we try to optimise both U and Z     - Instead we alternate repeatedly between fixing one and solving the other (coordinate descent)   + Solution on slide 19 and 20   **Autoencoder**   * Feed-forward neural network that tries to copy the input   + Contains a bottleneck layer - a hidden layer with dimensions much smaller than the input   + The bottleneck layer forces the model to compress and decompress the input with as little loss as possible * Autoencoders are another example of a dimensionality reduction model that we can apply to solve the matrix completion problem * **Linear autoencoders**    + Computes a U and Z such that     - Z is the compressed features and U is the decoder used to decompress Z to get the reconstructed output     - since the reconstructed output is to be as close to the input as possible   + This is a linear transformation in a linear autoencoder     - This means that the best linear autoencoder can only be as good as PCA * **Non-linear autoencoders**   + Uses a non-linear activation function, allowing it to be much more powerful than linear autoencoders   + Maps onto a non-linear manifold rather than a subspace |
| --- |



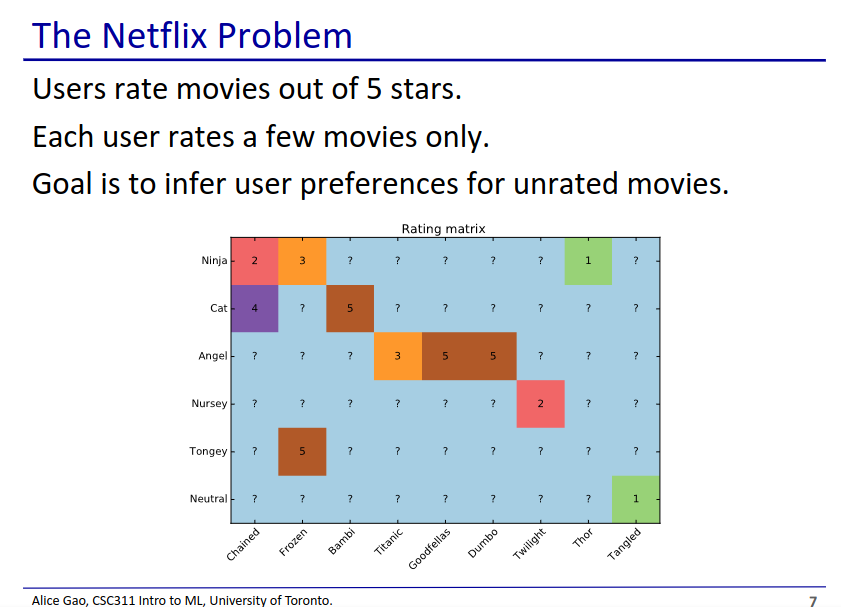




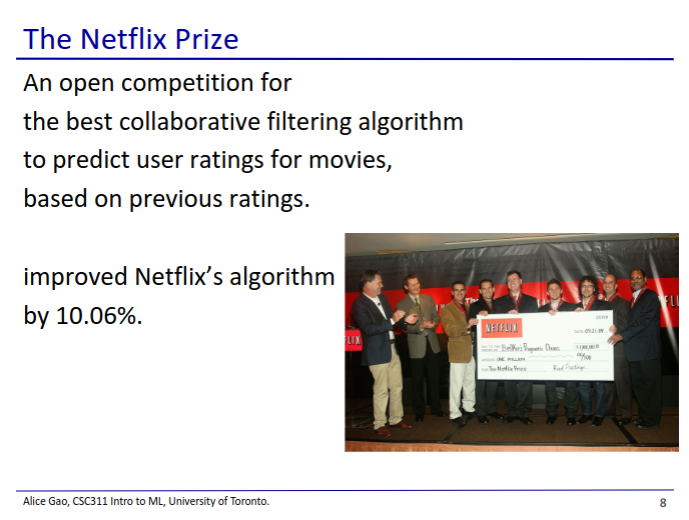




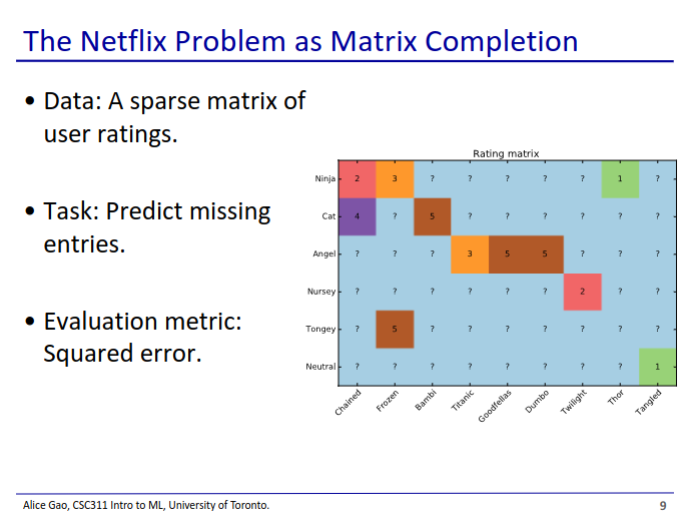
* Algorithms that try to figure out what things we might want to buy/watch
* Netflix uses your watch history to try to figure out what you might want to watch next



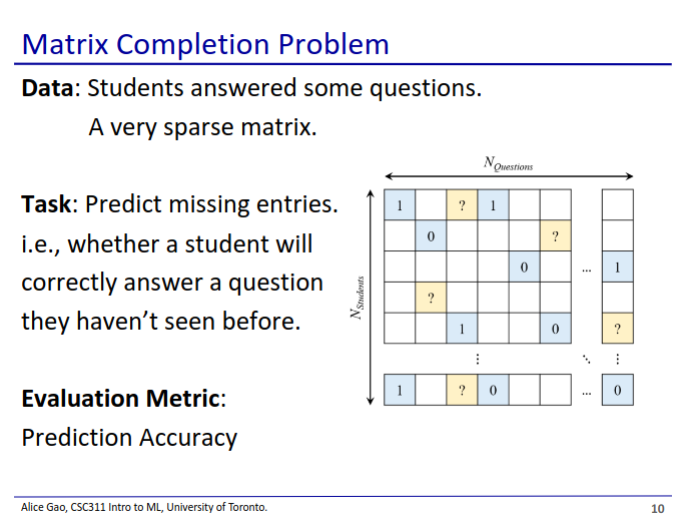
* Since the user cannot possibly rate every movie, the goal is to try to use the movies the user has rated to predict how they might rate other movies



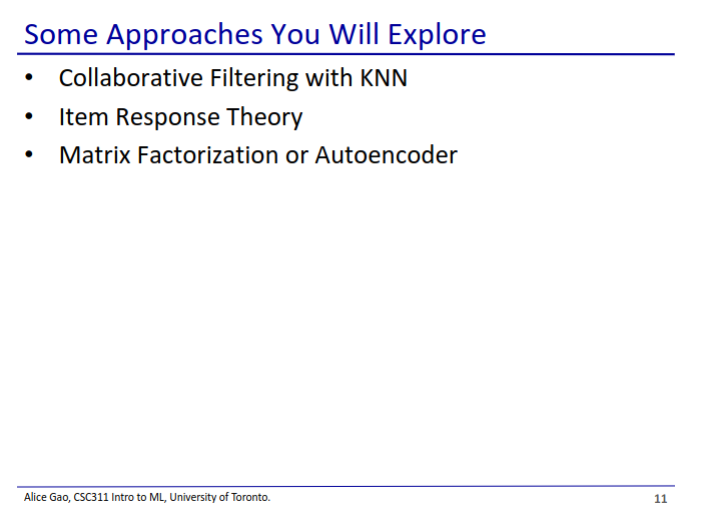
* Netflix hosted a competition to improve Netflix’s algorithm’s RMSE by 10%
* Took about 3 years to complete
* Finished by a team of scientists from AT&T
* Netflix got lucky with their 10% target
  + Later analysis of the data suggested that if the goal was higher, it may not have been possible
  + Shows the issues with working with messy real world data



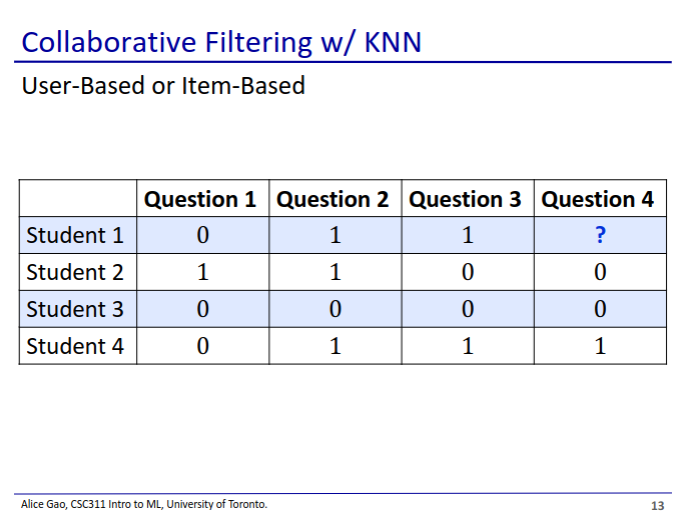
* This problem is a matrix completion problem
* Once the matrix is complete, Netflix can use the matrix values to make recommendations to users



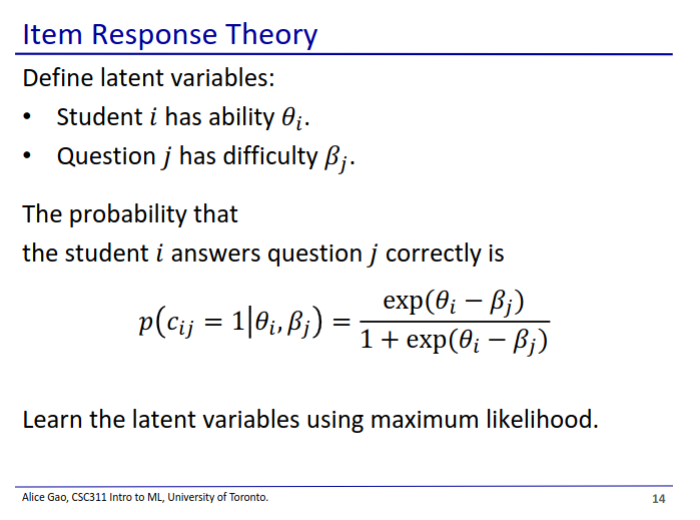
* This is the problem we will have on the course project
* This matrix is simpler since it is binary (1 for correct, 0 for incorrect)
  + We can use a simpler loss method: prediction accuracy



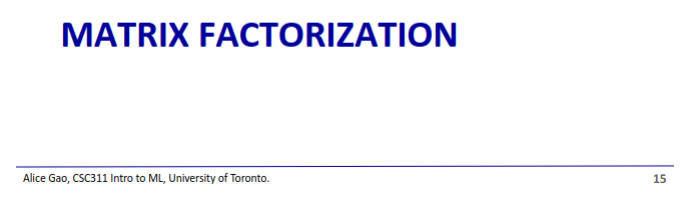


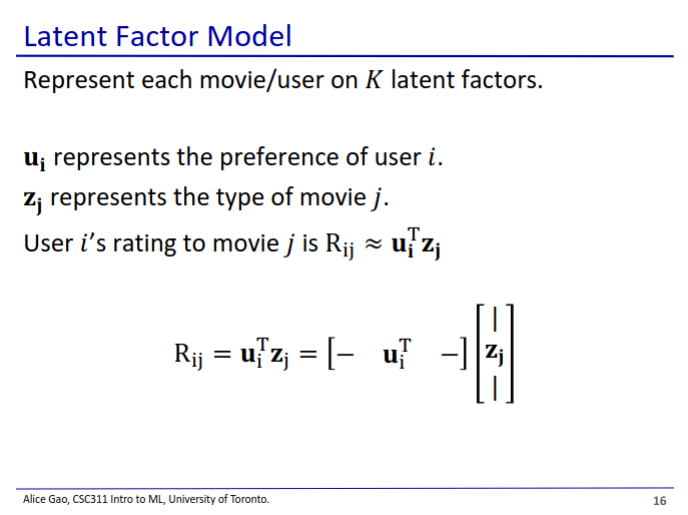


* Collaborative filtering is an ancient algorithm
* We want to predict whether student 1 will answer question 4 correctly
* User-based
  + We look at which other students answered other questions similarly to student 1
  + We then make student 1’s answer the same as the student that is most similar to student 1
    - Here we used 1NN, but we can also use KNN
  + Thus we would model student 1 after student 4
    - We predict 1
* Item-based
  + We look at which questions were answered by students similarly to question 4
  + We then make student 1’s answer to question 4 the same as student 1’s answer to similar questions
  + In this case Q3 is most similar to Q4. So we model Q4 after Q3
    - So we predict 1

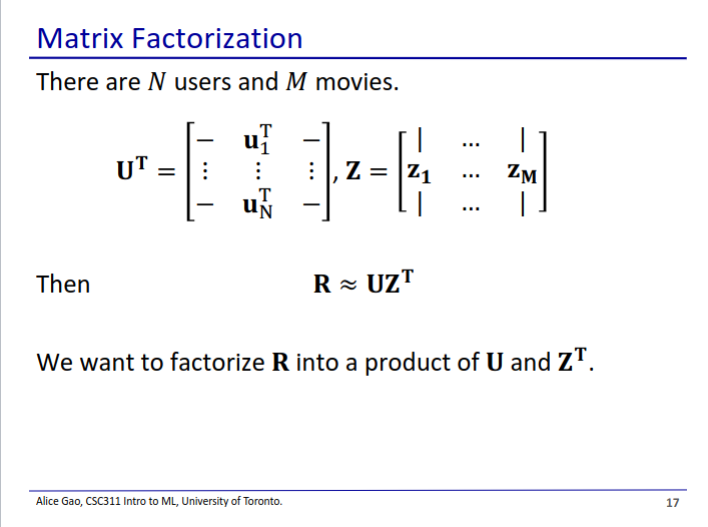


* We have some latent variables that are part of the model, but that we don’t observe
  + Each student has an ability score
  + Each question has a difficulty score
* The probability that a student answers a question correctly is a function of the student’s ability and the question’s difficulty
  + Note that this function looks like the sigmoid activation function in logistic regression
* Sanity check: Make sure that when ability is higher probability is higher, and when difficulty is higher probability is lower
* We then find the student’s ability and question’s difficulty by taking the derivative of the log-likelihood

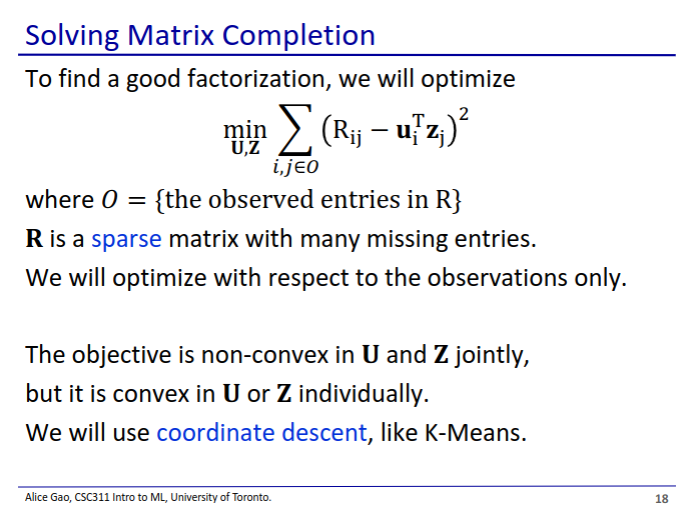




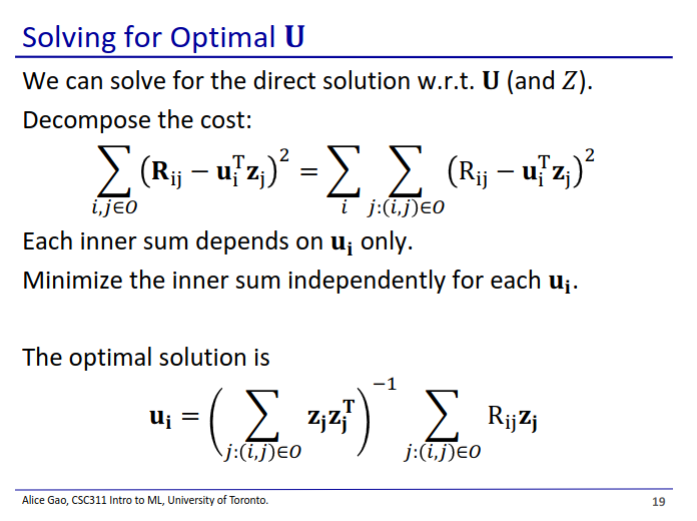
* Matrix factorisation is a dimension reduction model, but works for this problem
* We have 1 vector for each user, and 1 vector for each item
  + Vectors on a reduced dimensionality subspace of K latent factors
    - Vector features represents how well an item falls into each category
    - User features represents how well a user likes each category
  + Taking the dot product of the vectors gives the rating for the movie



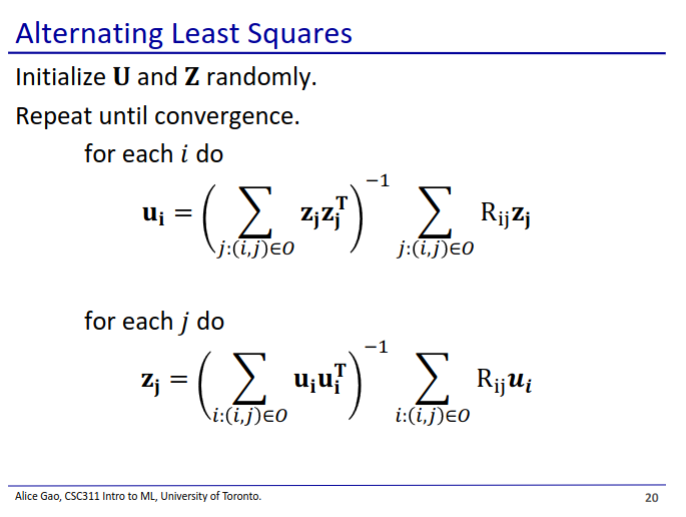
* We are given the ratings R matrix, but this is a sparse matrix that is missing some values
  + We want to train the UZT matrix such that its values match the ones we have in R
* We do this by factorising R into U and ZT
  + Some of the dimensions of U and Z are smaller than that of R
  + U has dimensions (N, K) and Z have dimensions (M, K)
    - We define what K is (hence dimensionality reduction)



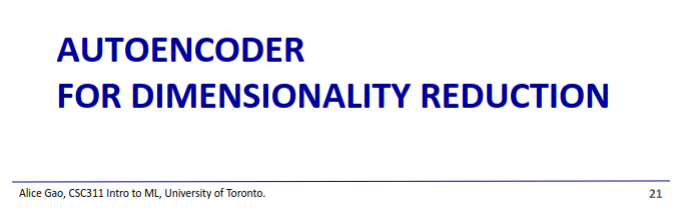
* We can only optimise with respect to the values we have in R
* This is very hard if we try to learn both U and Z at the same time
  + Instead we can fix either U or Z and learn the other

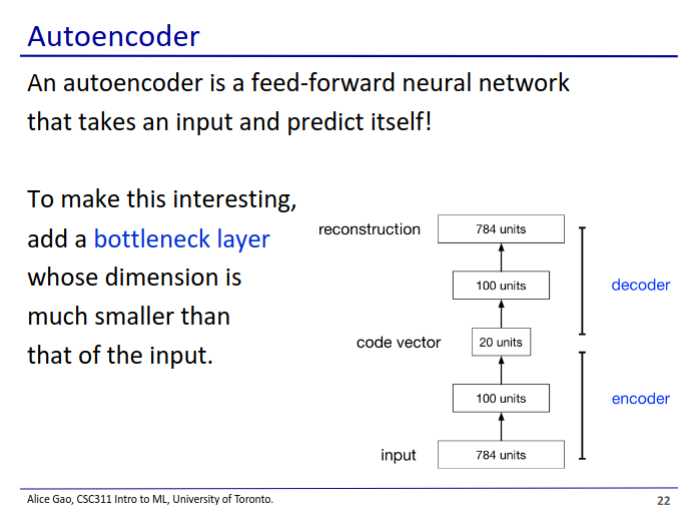


* Here we fix Z, allowing us to solve for an optimal U

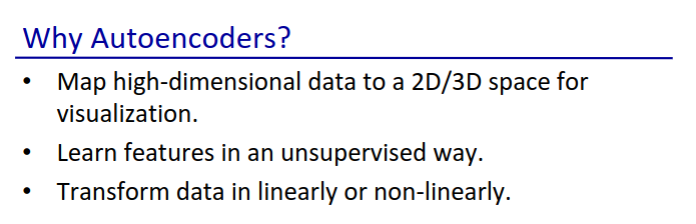


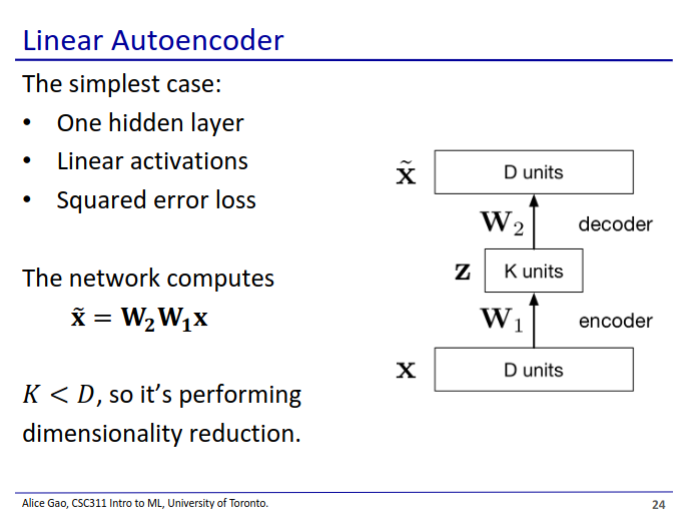
* We can alternate between solving for U and Z repeatedly to get a good answer for both



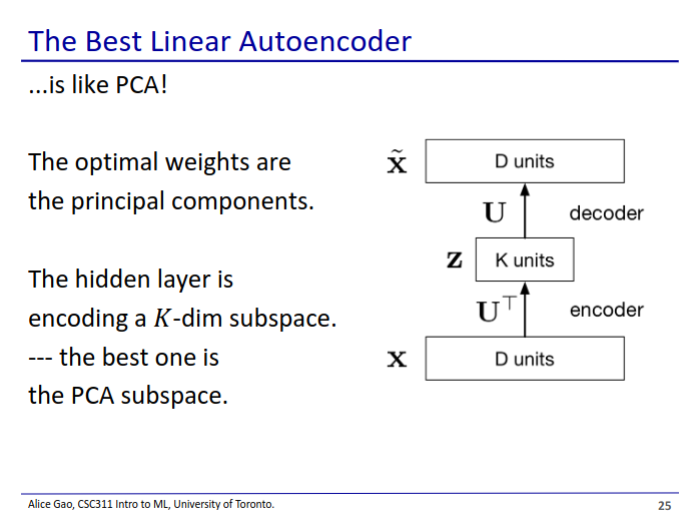


* Produces an output with the same dimensions as the input
  + Tries to reconstruct the data that it was given as accurately as possible
* The middle layers are much smaller than the outer layers
  + This forces the model to compress and then decompress the data without losing too much information

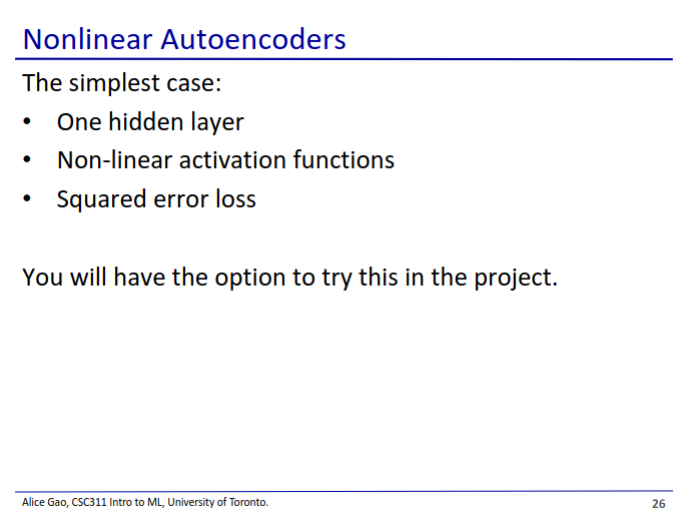


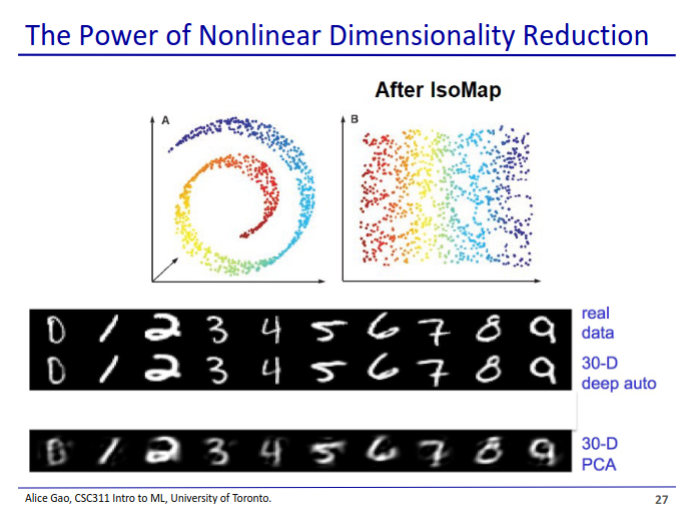


* We have linear autoencoders and non-linear autoencoders
  + Difference is based off of what activation function we use
  + Linear autoencoders have linear activation functions, non-linear autoencoders have non-linear activation functions

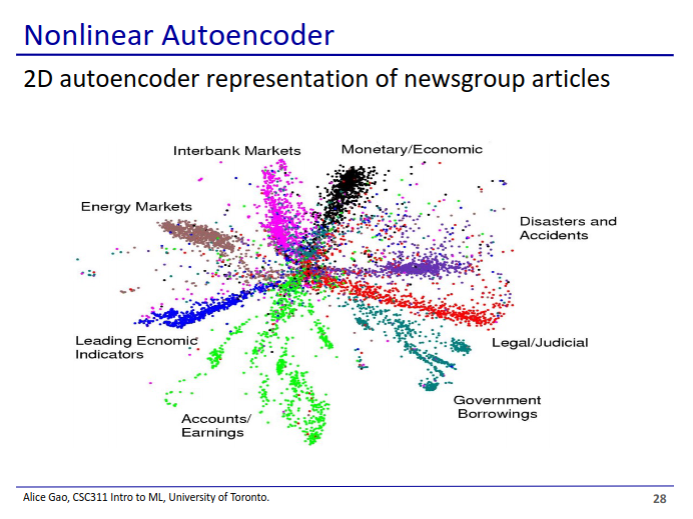


* A linear autoencoder can only be as good as PCA
  + PCA is a linear model, which we proved in previous lectures to be ideal
* Thus there is really not much of a point in doing a linear autoencoder since linear neural networks can only do linear transformations





* The images reconstructed using a non-linear autoencoder are much sharper than PCA
  + Suggests it does a much better job at compressing and reconstructing the data than PCA



* Every dot represents a news article and colours are the clusters