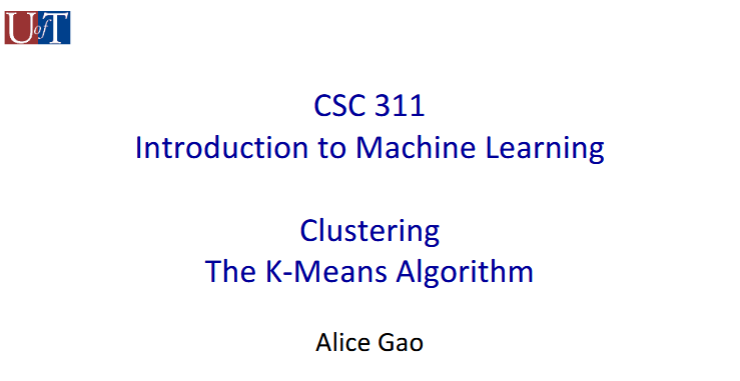
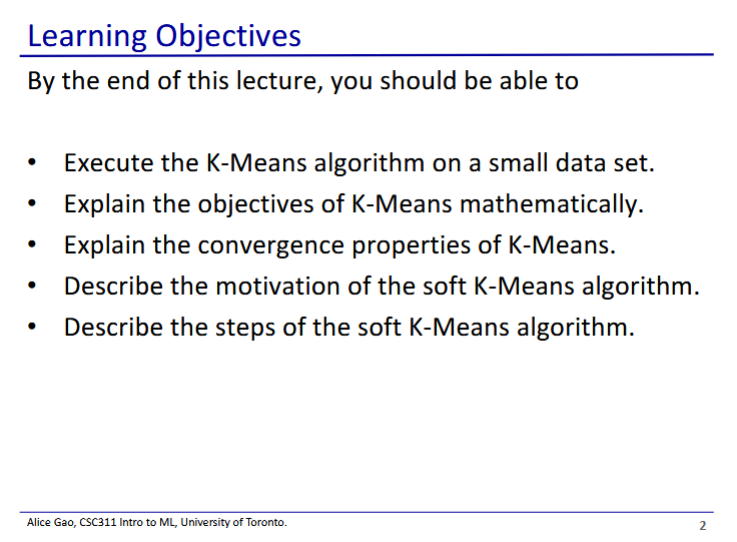
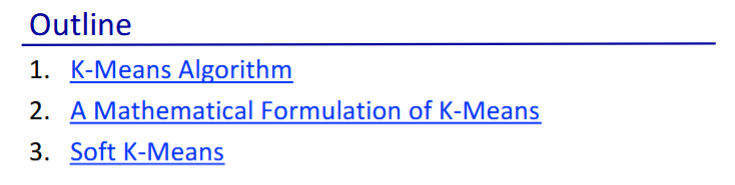
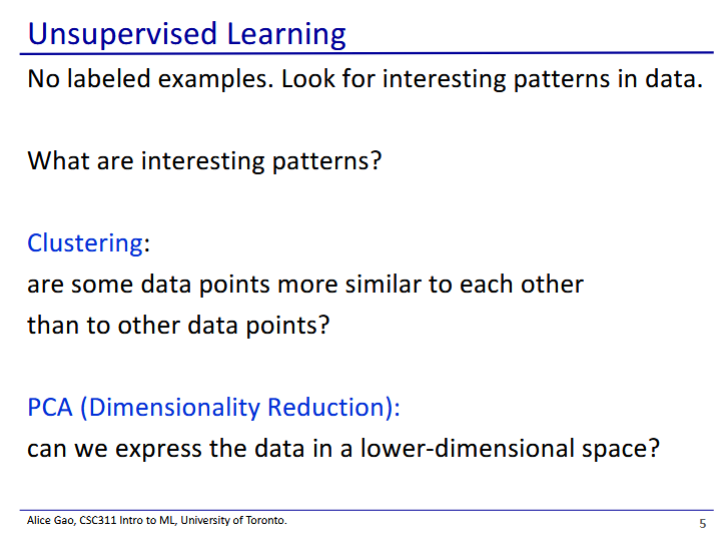
| Unsupervised learning  K-means algorithm   * Minimize square distance between points and their assigned cluster centers * K-means prefers clusters that are spheres   K-means algorithm (theory) |
| --- |



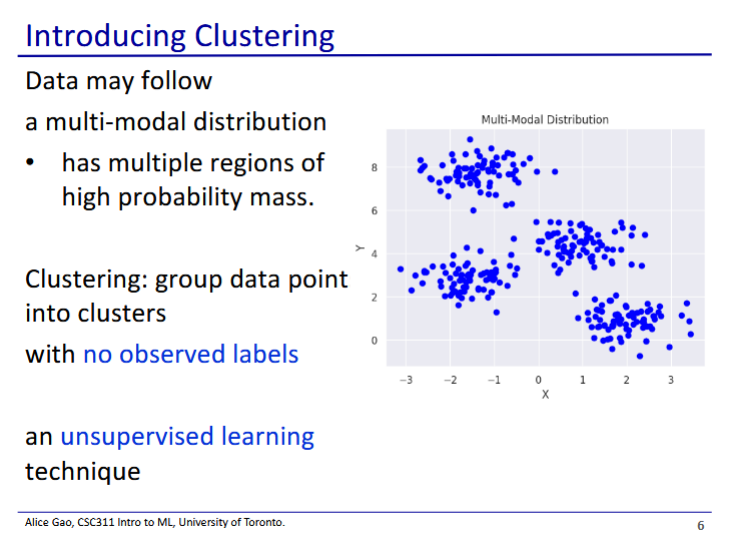




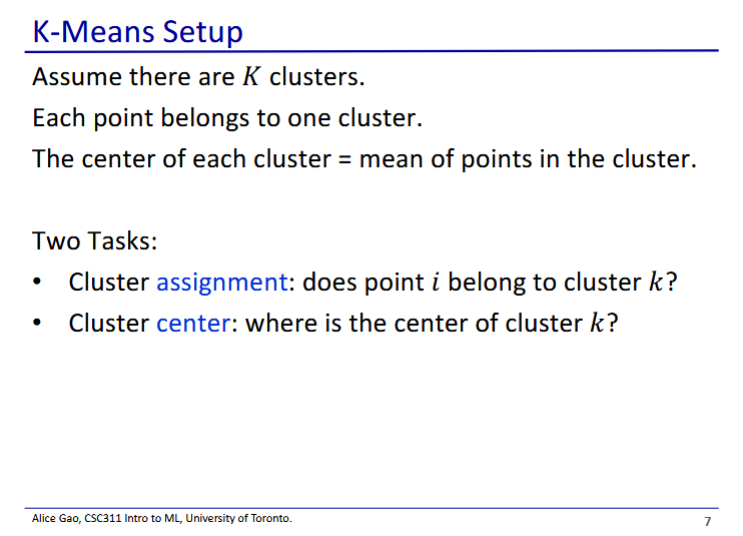




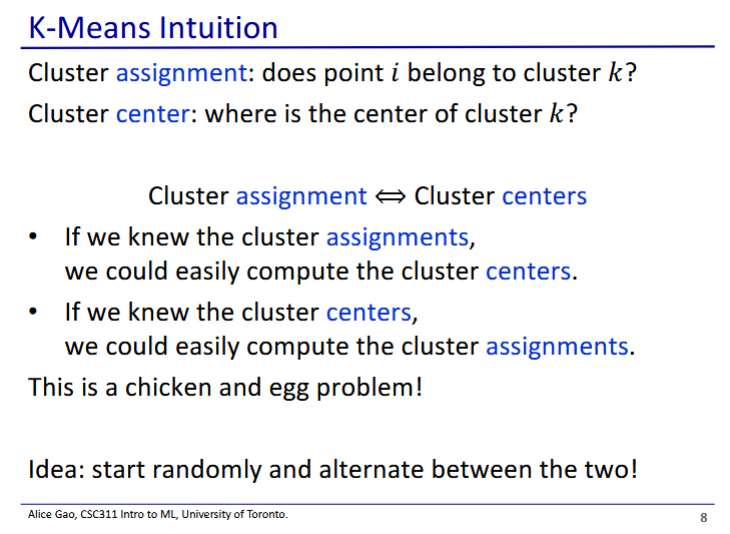
* In supervised learning we have data with labels
* In unsupervised learning, we only have data an no labels
  + Our goal is to look for “interesting patterns”
  + This week we will look at clustering - grouping together data points that are more similar than the other
  + Next week we will look at dimensionality reduction



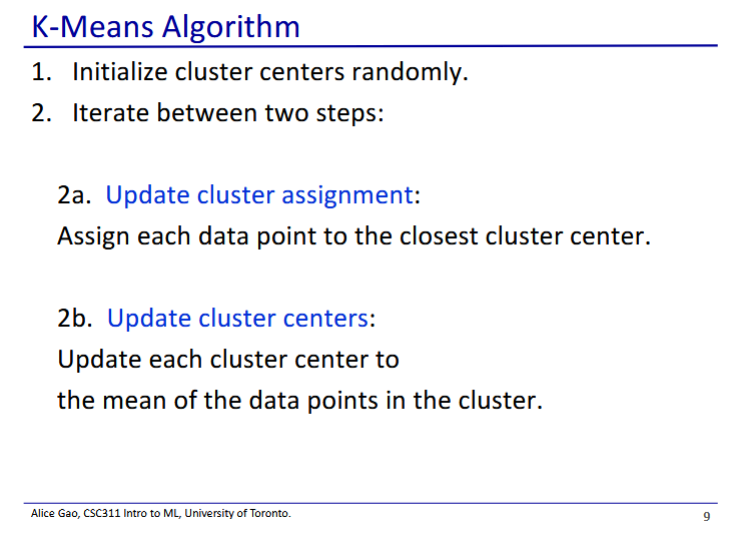
* Clustering is to group the data points into clusters
  + The data may follow a multi-modal distribution where some regions have a higher probability mass of points than others
  + We want to group our data by these regions without using labels
* A motivating example
  + Grouping scientific papers into different topics without reading them or labelling them



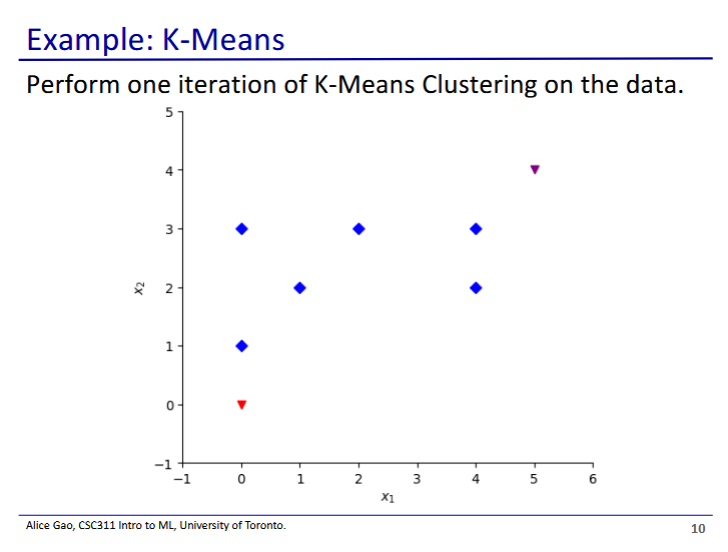
* In K-means we assume that the data has K clusters
  + K is a hyperparameter
* Once we have decided on the number of clusters we then have 2 tasks
  + Which points belong to which cluster
  + Where is each cluster centred?



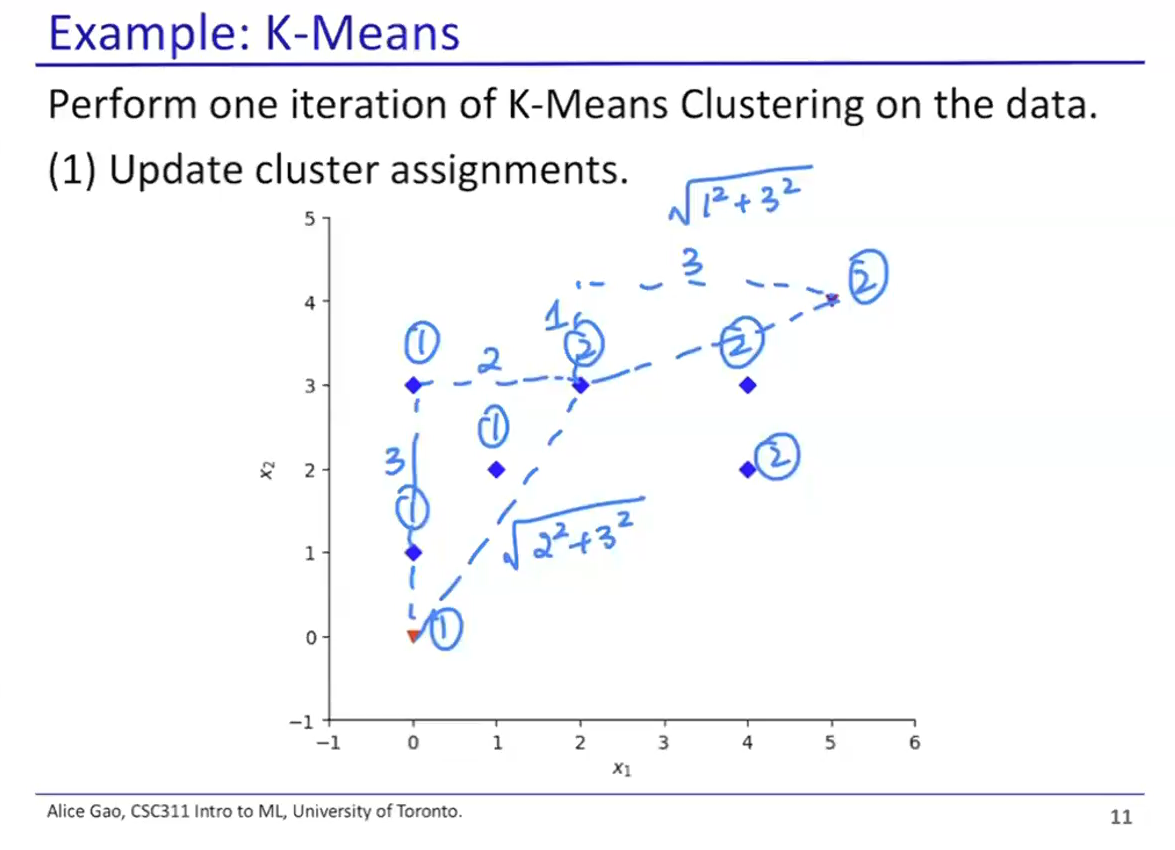
* There is an interesting relationship between these 2 tasks
  + If any one of them is fixed, the other one is simple
    - Suppose we already know the cluster assignments, it would be easy to figure out the cluster centres (find the center of all points in the cluster)
    - If we know the cluster centers, it is easy to figure out which points belong to the cluster (assign to closest cluster0
* Thus we can try alternating between these 2 tasks



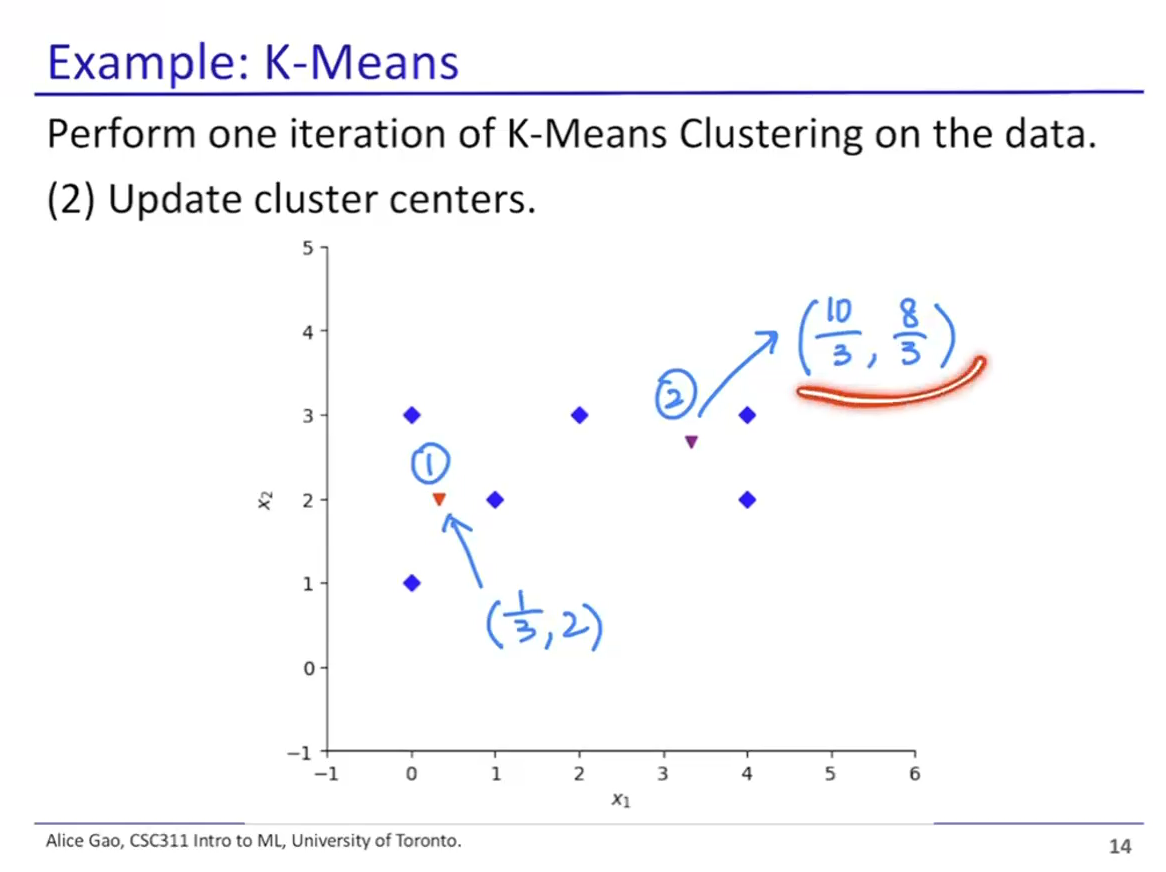
* We repeat steps 2a and 2b until a convergence criteria (to be mentioned later)



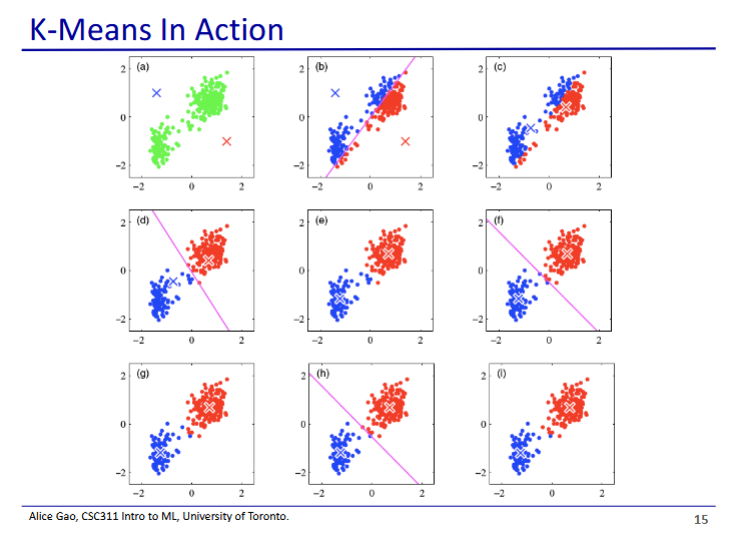
* Data has 6 points
* Cluster centres are shown in orange and purple
  + Orange: (0,0)
  + Purple: (5, 4)
* Each iteration has 2 steps
  + Update cluster assignments
  + Update cluster centers



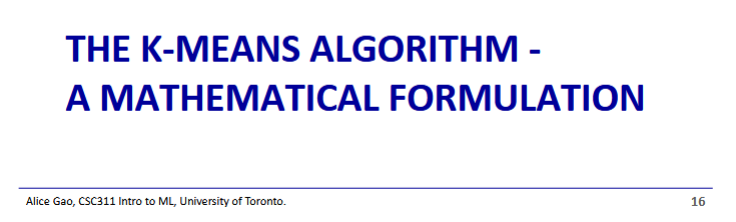
* We assign points to clusters based on which cluster center is closest

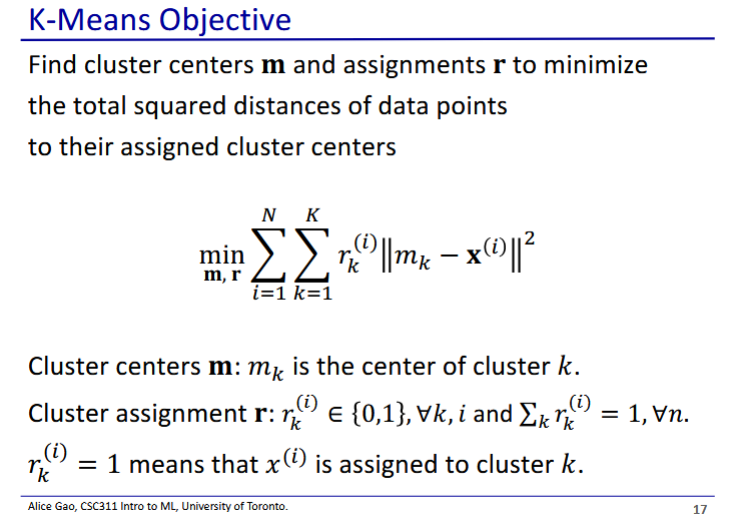


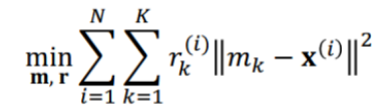
* Calculation:
  + Average out the x and y values for each point in the cluster
* Orange:
* Purple:
* What happens if we continue the k-means algorithm?
  + In this case nothing else would happen
  + The cluster assignments did not change when we updated cluster centers
  + This is our convergence criteria: if our cluster assignments does not change, we are done

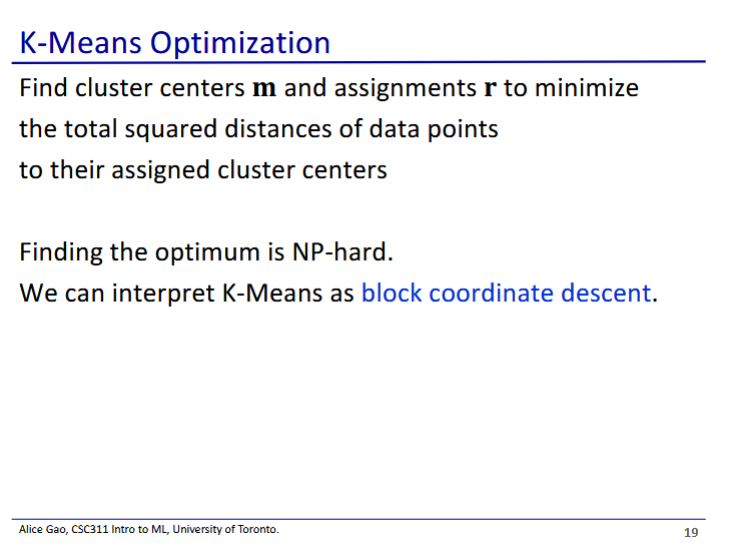


* This shows k-means in action
* The cluster centers slowly migrate over to each group of points
* Images alternate between cluster assignment and cluster centers
* <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>
  + Website for visualising k-means algorithm in action
* How do we decide which K to use?
  + We typically use a validation set to check how many cluster centers we need

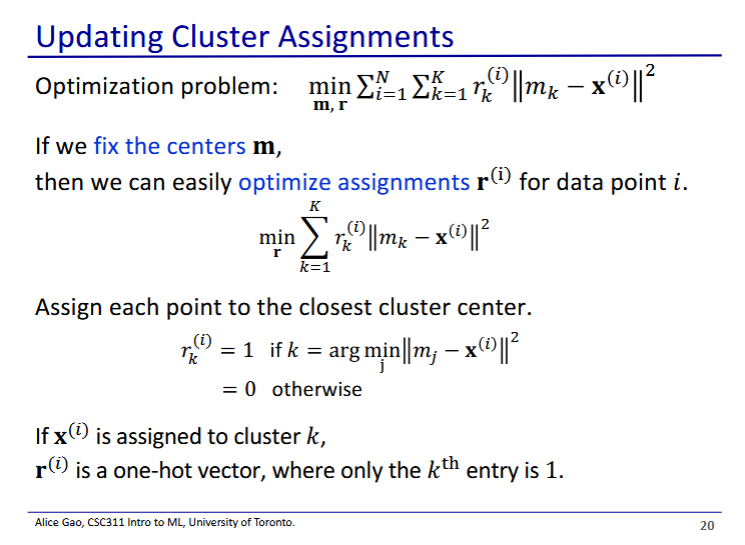




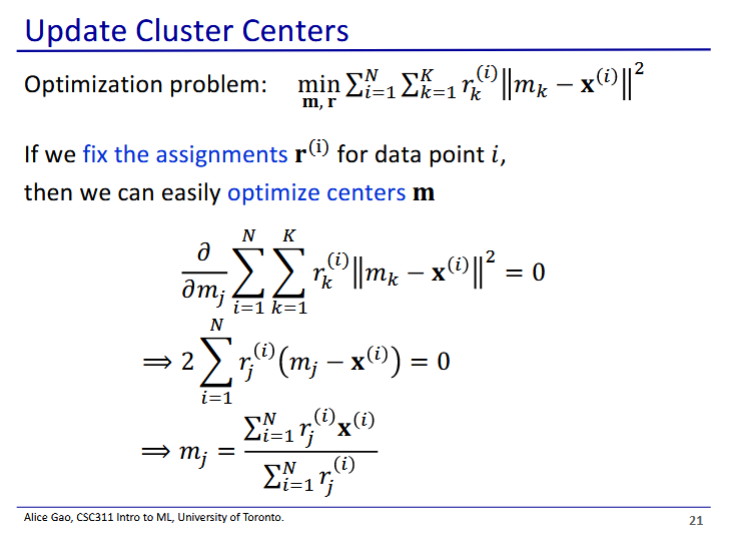
* m denotes the cluster centers, r denotes the cluster assignments
  + m is a vector, r is a matrix
* Cluster centers m: is the center for cluster k
* Cluster assignment r: is either 0 or 1, and denotes if i is in cluster k
  + If we sum up , we should always get 1
    - Since data point i can only be in 1 cluster
* Goal is to minimize square distance from data points to cluster centers
  + 
  + Key term is , which is only 1 if the data point belongs to the cluster being examined
  + Alternatively: objective is to minimize the variance within each cluster



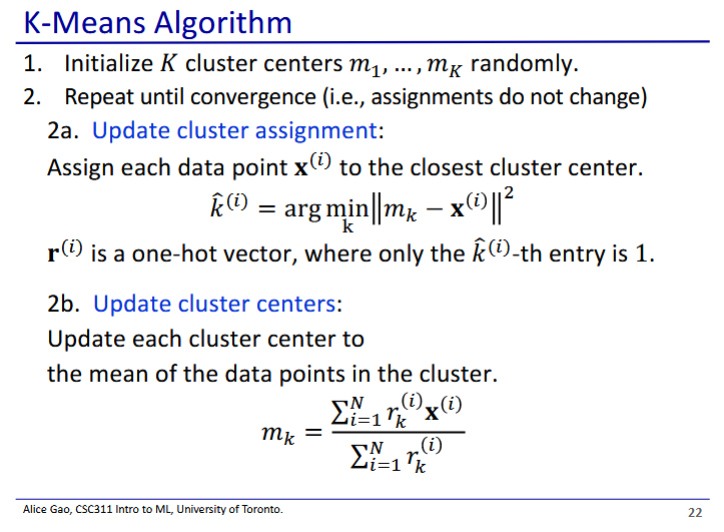
* Finding which clusters and assignments directly is hard
* Instead we can do this iteratively through block coordinate descent
  + Block means the algorithm changes multiple variables in one step
    - In K-means it means we change the cluster assignments in one step, and then the cluster centers

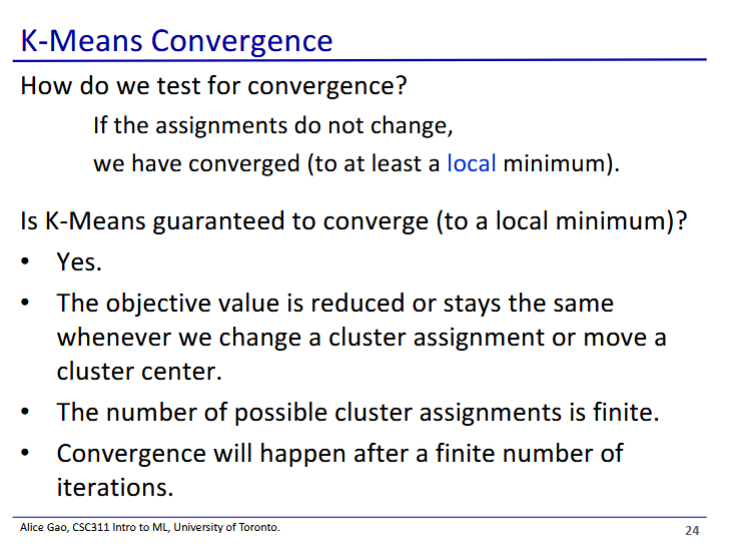


* Algorithm has 2 parts: updating cluster assignments and updating cluster centers
* If we fix the centers, we can easily optimize assignments
  + To get the assignment we find the cluster that results in the minimum square distance for our point

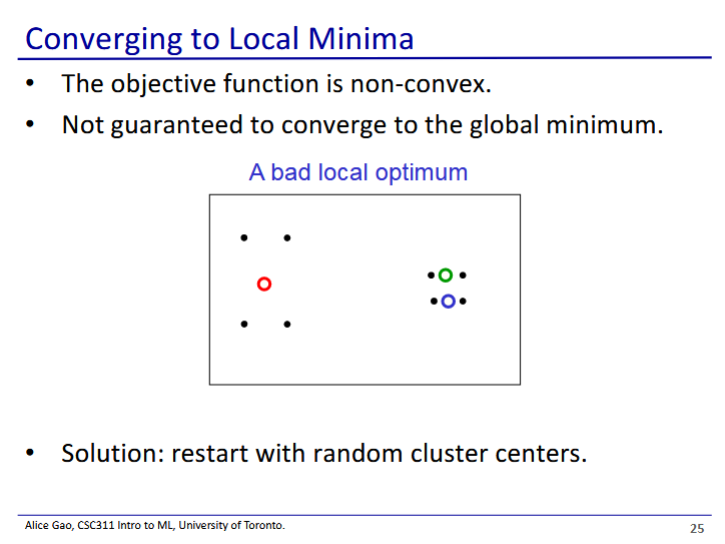


* To optimize the cluster center we take the derivative with respect to each cluster center

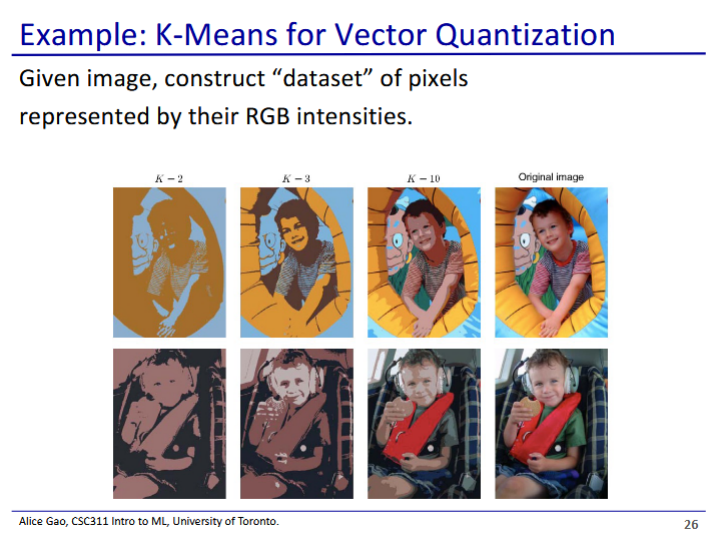




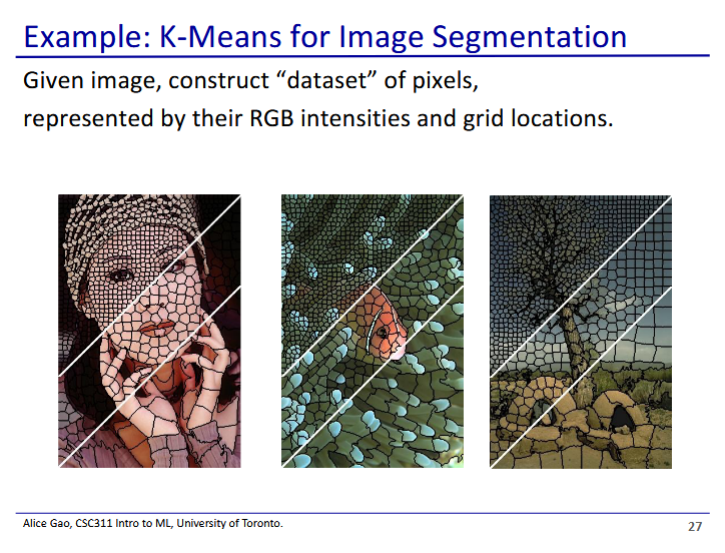
* K-means is guaranteed to converge to a local minimum - it will stop somewhere eventually
  + However where it ends up might not be the most optimised cluster result (global minimum)
  + Proof:
    - Objective value is always reduced or stays the same when we update either cluster assignments or cluster centers
    - The number of possible cluster assignments is finite
    - Thus K-means will always stop somewhere since both values are finite (in the worst case we iterate through all of them)
* One way to evaluate our clusters would be to use our objective function and see if distance is a small enough value for our liking



* We could get stuck at a local minimum that isn’t what we want
  + K-means objective function isn’t very complex
* The solution in practice is to restart with new random cluster centers
  + This works for gradient descent too if it gets stuck at a local minimum



* Using K-means to cluster the pixels into K colour groups
  + We then recolour pixels by their group average to get a cool image effect



* Here we are clustering both by colour value and by location
  + Thus pixels are clustered if they have a similar colour and location
  + Divides the picture into “super pixels”