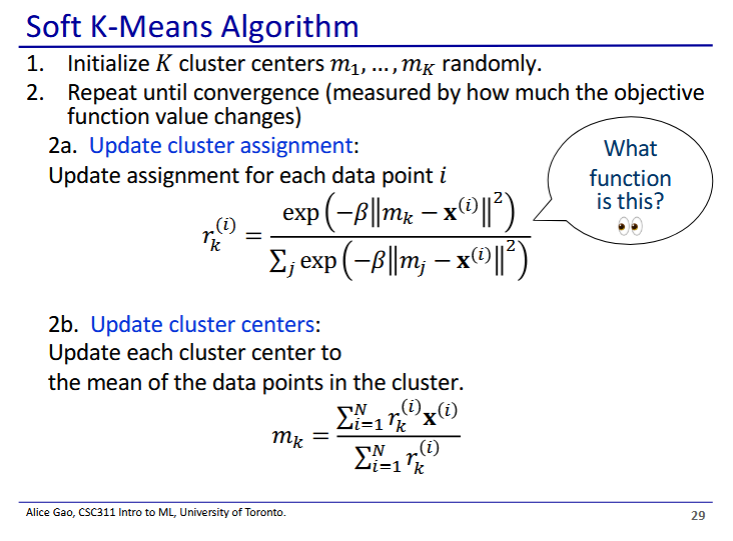
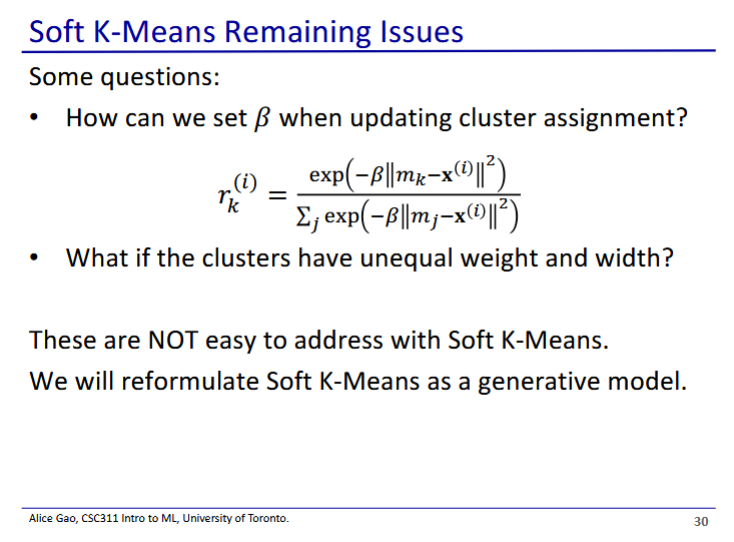


* Regular K-means makes hard cluster assignments - each point can only belong to one cluster
  + Might not work very well with noisy data that does not fit perfectly into clusters
  + is a one-hot vector
* Soft K-means makes soft cluster assignments - we assign points to clusters via probability
  + is a vector of K probabilities

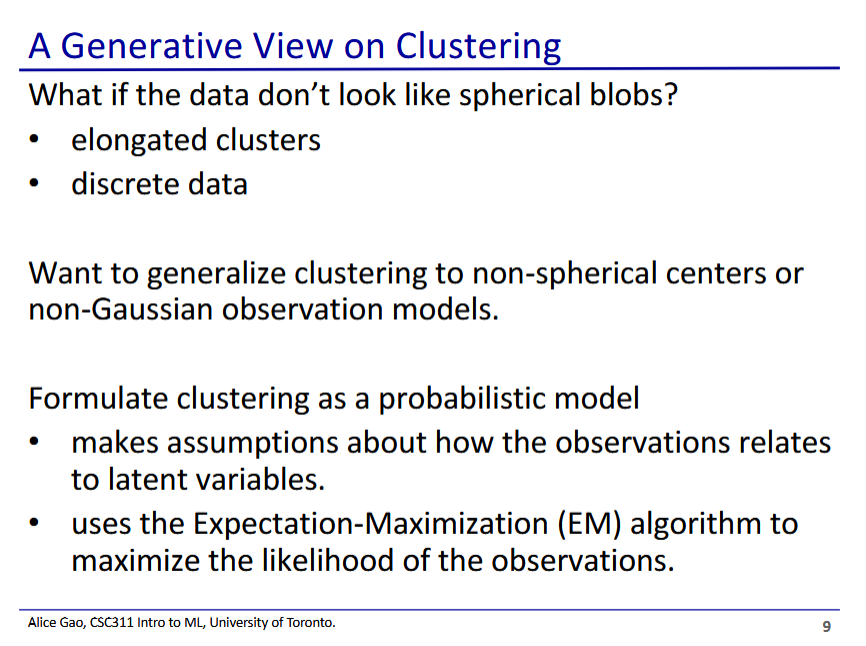


* Steps 1, 2, and 2b are the same as for hard K-means
* Only changed step is 2a
  + Now we have a probability for cluster assignment
  + This function is the softmax algorithm
    - Converts cluster assignments to proportional probabilities

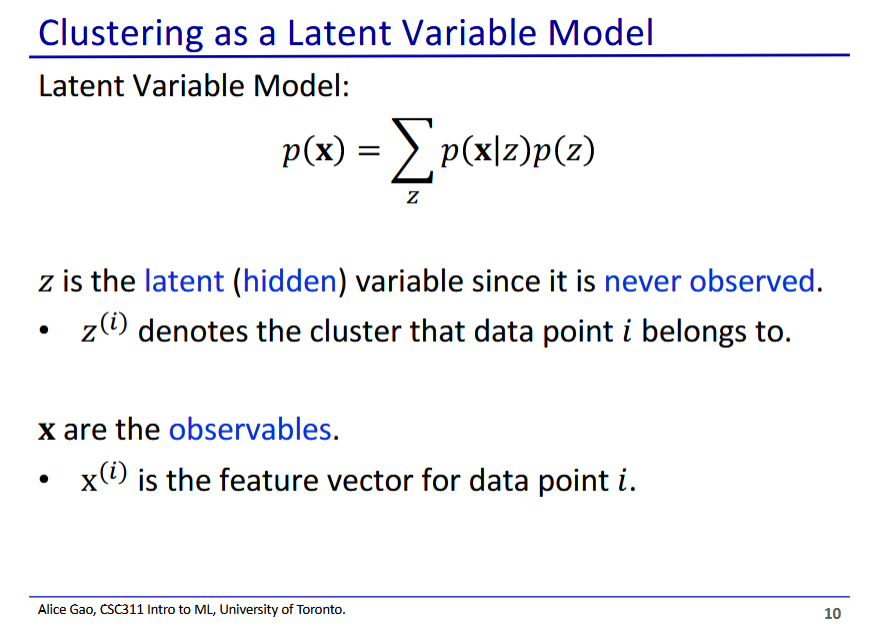


* Soft K-means doesn’t really work well in practice
  + How do we set beta?
  + What if clusters have different shapes?
* Thus in practice we will use other models for soft cluster assignments (Gaussian mixture models)

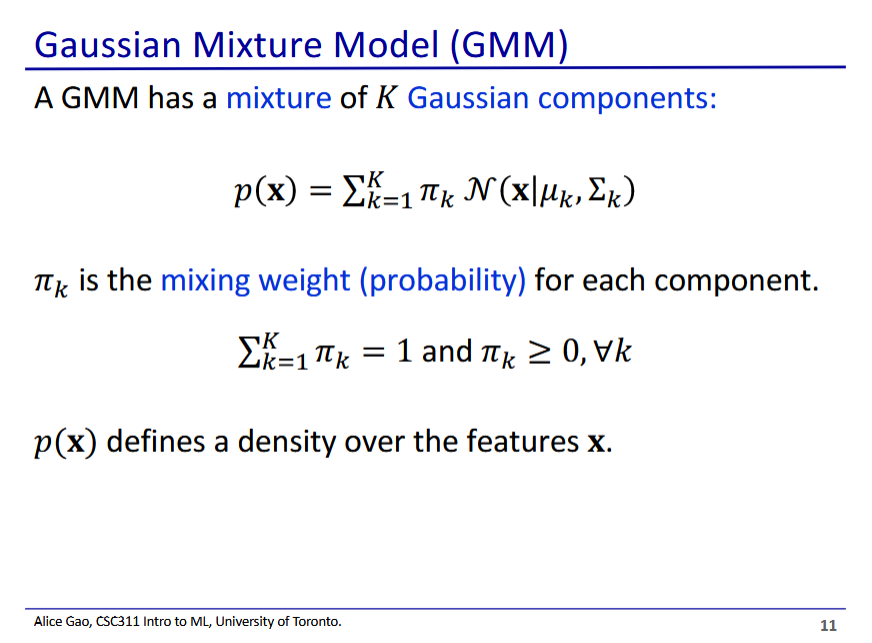




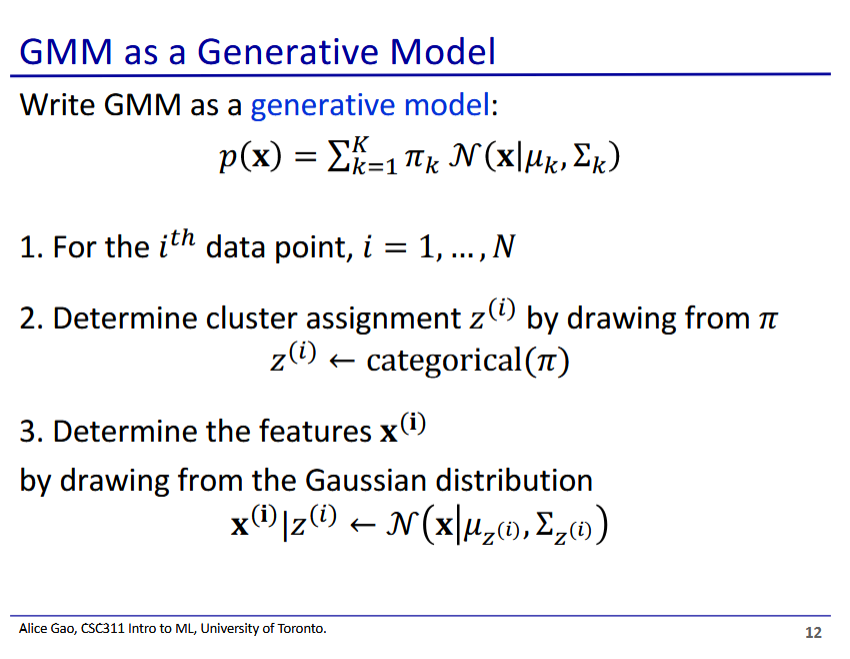
* With regular K-means, there is a bias towards spherical clusters due to how the objective works
  + Clusters in our data may not always look like this, so we need a way to account for this



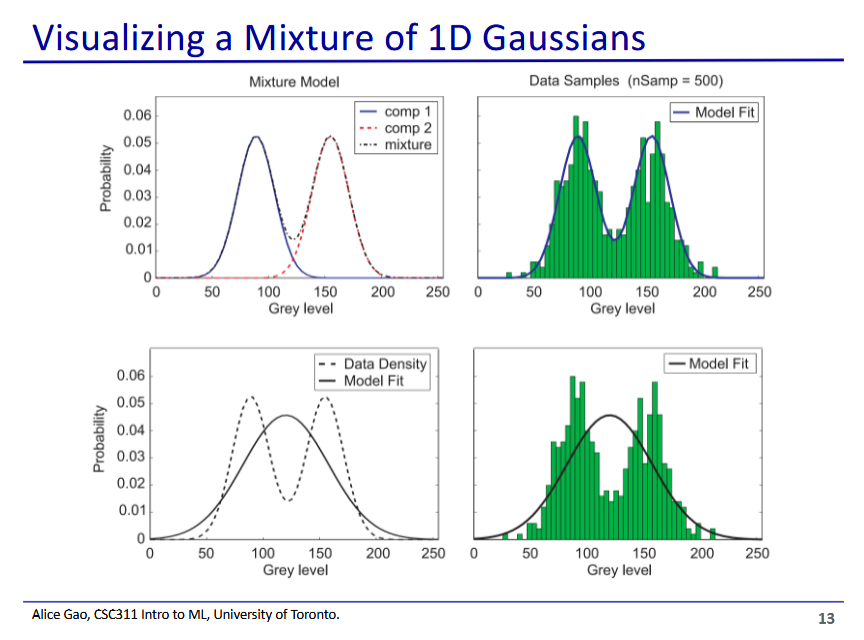
* Gaussian mixture model is a latent variable model
  + The model contains an internal hidden variable that is updated by the model but never observed by us
* Contrast z with x
  + x are the observables, which is part of the data
  + z is a latent variable, it is not part of the data and is predicted by the model
    - denotes what cluster data point i belongs to
* p(z) is kind of the prior for the distribution



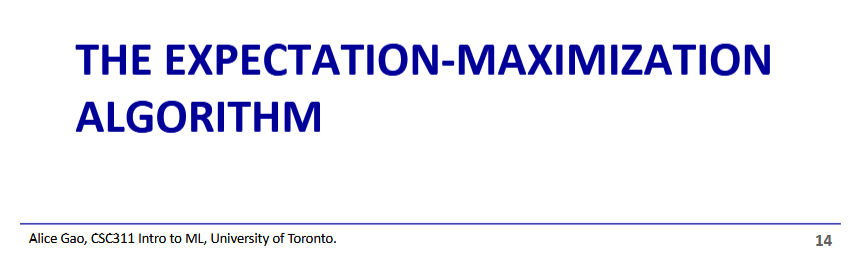
* We have K gaussian distributions, representing the k clusters
* We also have a mixing weight , which all sum to 1
  + This is the prior for each cluster (what proportion of training data is in each cluster)

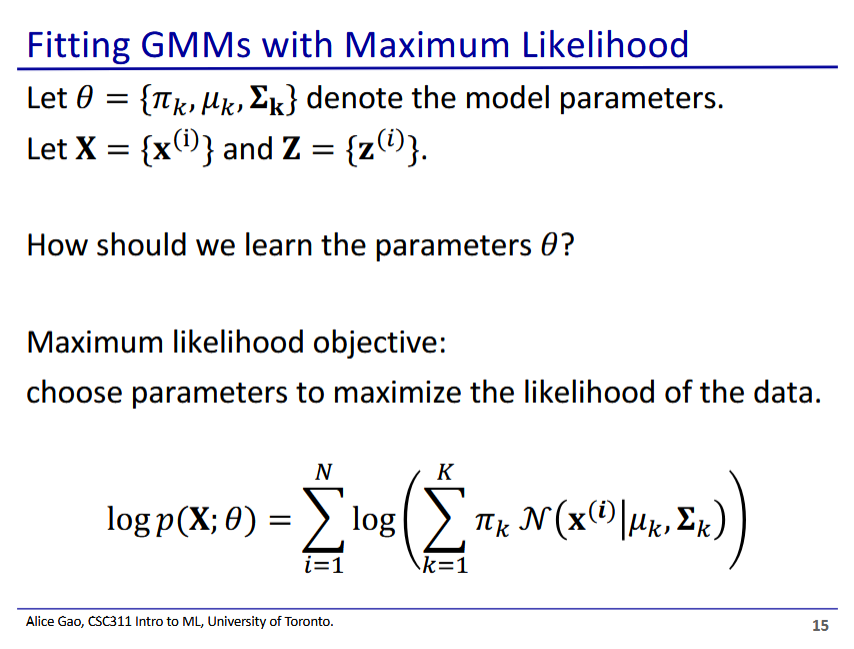


* We can think of GMM as a generative model that generates data
* Suppose we generate N data points
  + First we determine cluster assignments for each point by drawing from
    - This is a categorical value to get a single cluster the data belongs to
  + Then we determine the feature
    - The chosen cluster follows a Gaussian distribution, so we can determine the features that follows the distribution

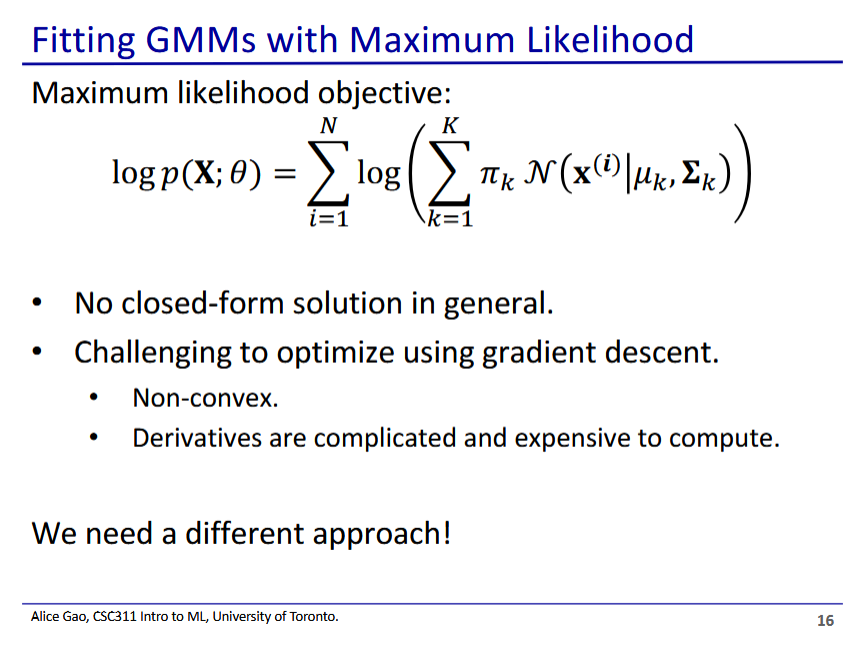


* Green bars represents the data, and we are trying to fit a 1D gaussian model
* Top: fitting using 2 Gaussian distributions
* Bottom: fitting using 1 Gaussian distribution
  + 1 distribution is not enough to model the data

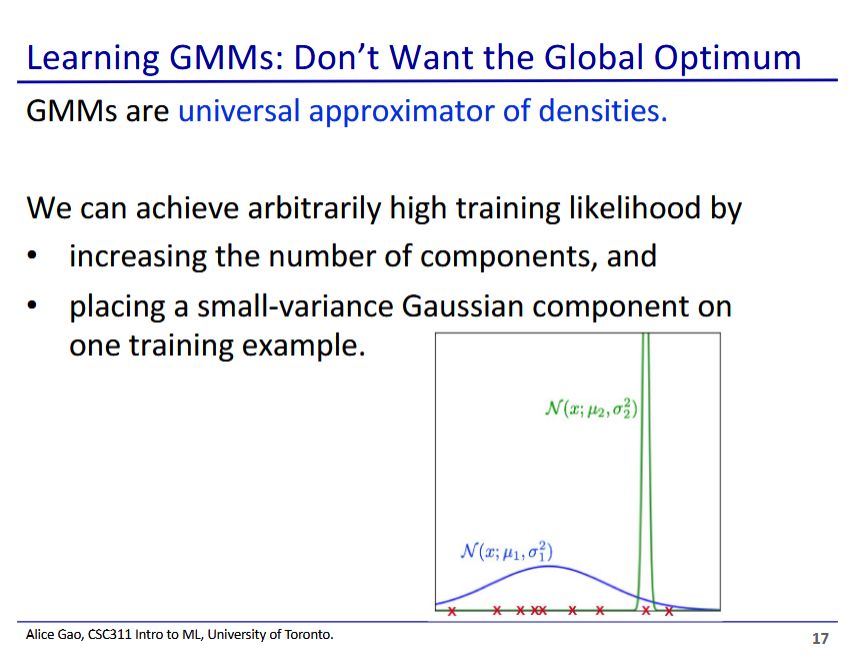




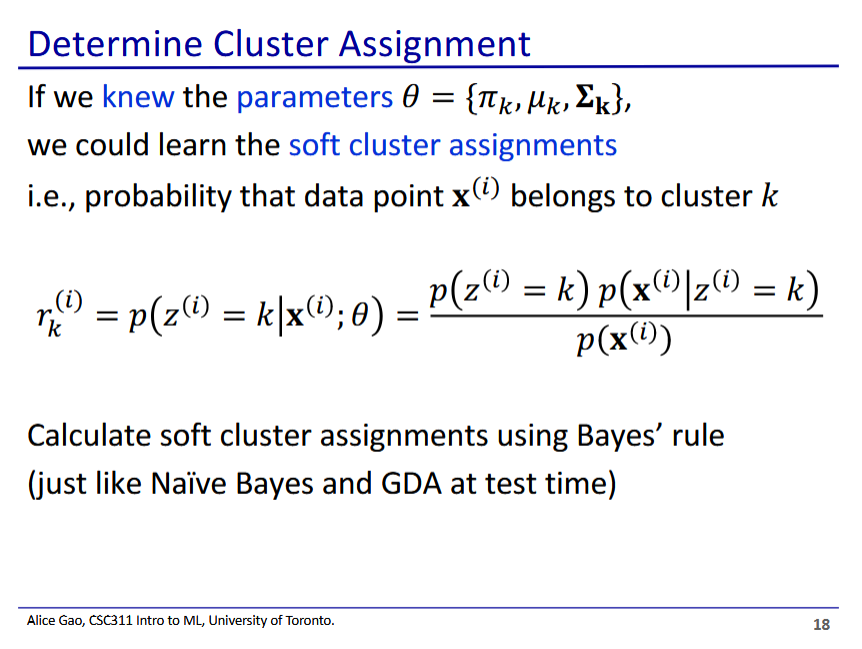
* There are 3 parameters for the model
  + mixing probability for the Gaussian components
  + is the mean for the Gaussian components
  + is the covariance matrix for the Gaussian components
* Then we have some other quantities:
  + X is the training data
  + Z is cluster membership (latent variable) we are trying to estimate
* We can use maximum likelihood to learn our parameters
  + To do this we need to derive the log-likelihood (shown above)
  + Sum over all points
    - Sum over probability that is generated by distribution k



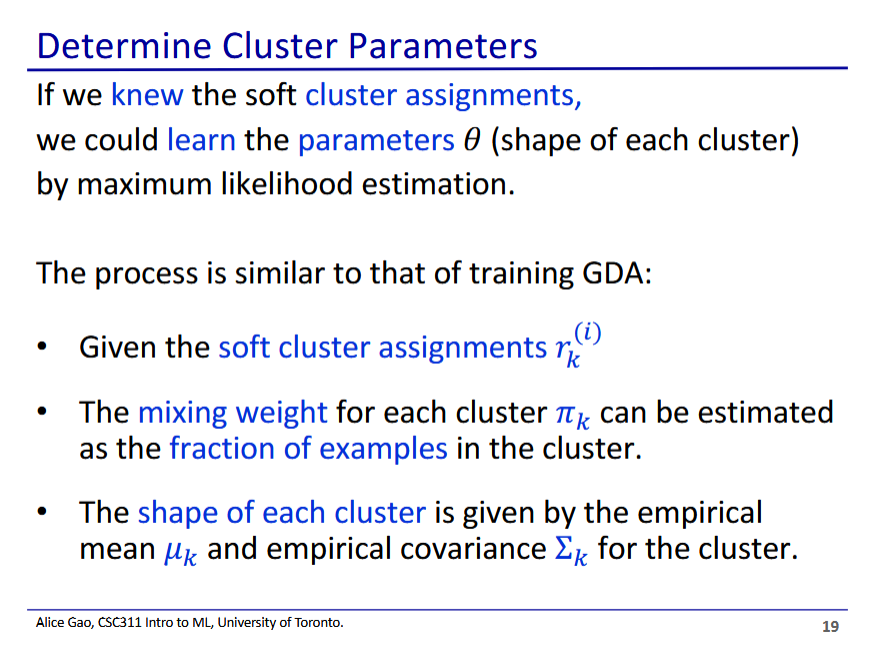
* We are not going to directly optimise maximum likelihood, since it is hard
* Thus we will do it iteratively in a similar way to K-means



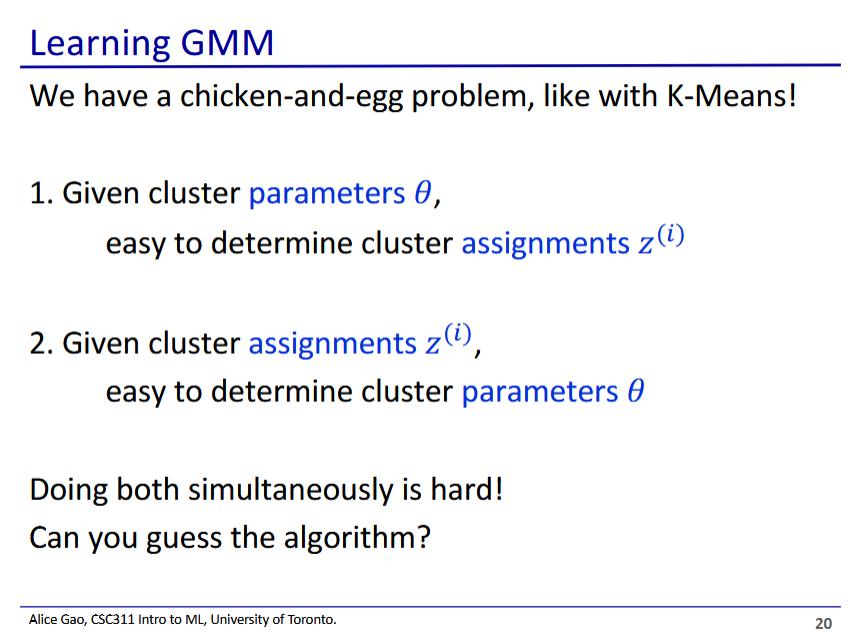
* Gaussian mixture models are very expressive: is a universal approximator of densities
  + Similar to neural networks or decision trees which can approximate any function arbitrarily well
  + GMMs can approximate any probability density function arbitrarily well
* Thus we need to be very careful to not overfit to the training data
  + We can fit a single example really well with something like the green line above
* Thus we may not want to actually reach the global maximum, since it may not produce a useful model



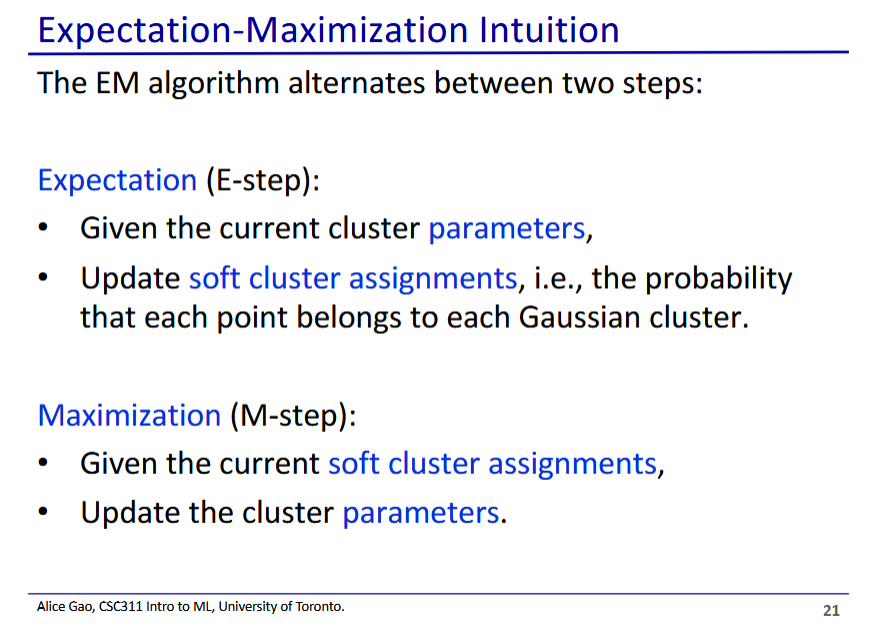
* We will solve for our parameters iteratively in a similar way as we did K-means
* Determine cluster assignment
  + Fix parameters (assume we know what the clusters look like) and use those to determine the soft cluster assignments
  + We determine the soft cluster assignments using Bayes’ rule
* Equation
  + is the probability that belongs in group k (posterior)
    - is a one-hot value, so i is either in k or it isn’t
    - But we don’t actually get to know , so the best we can get is a probability that is in k
  + We calculate the posterior using Bayes’ rule
    - is the prior, which is the same as
      * Without seeing any data, what is the probability that is in cluster k?
    - is the likelihood - we get this using the appropriate Gaussian probability density function ()
* Predicting cluster assignment is the same process as when we make a prediction using Naive bayes or GDA once the model is trained



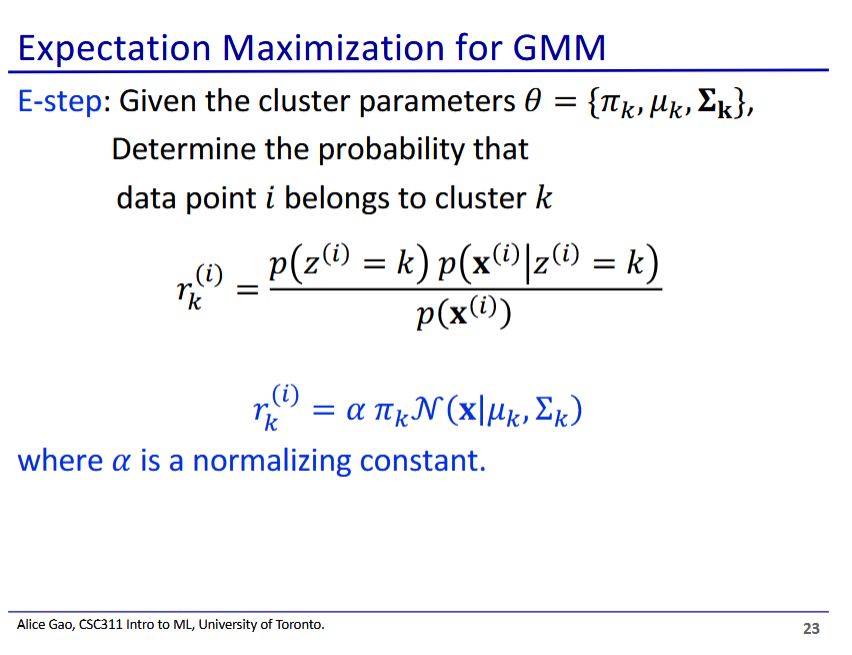
* After we assign points to clusters, we then fix the assignments and recalculate the distribution/cluster positions
* The formulas for this step look very similar to training GDA
  + We are learning Gaussian distributions, so it will look similar to GDA
  + Given our cluster assignments, what should our Gaussian distribution for this cluster look like?
    - Mixing weight is the fraction of total points in the cluster
    - Mean and covariance becomes the empirical mean and covariance of points in the cluster



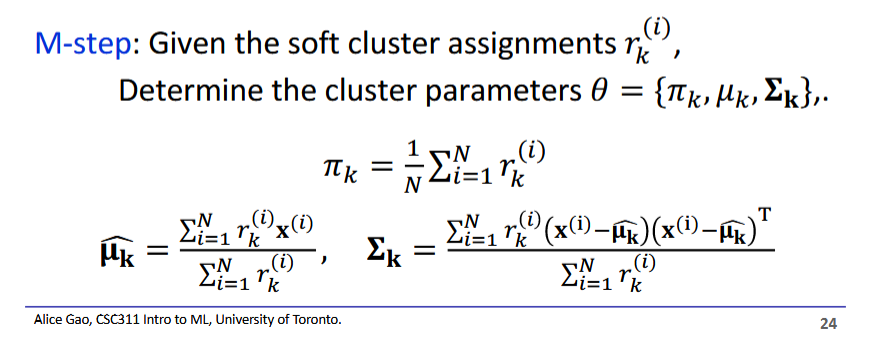
* Thus we have the same problem with K-means:
  + We need the clusters to easily determine the assignments
  + We need the assignments to easily determine the clusters
* The solution is the same: to repeatedly fix one and calculate the other until we reach convergence
* Typo on slide: we determine the soft cluster assignments and not the hard cluster assignments



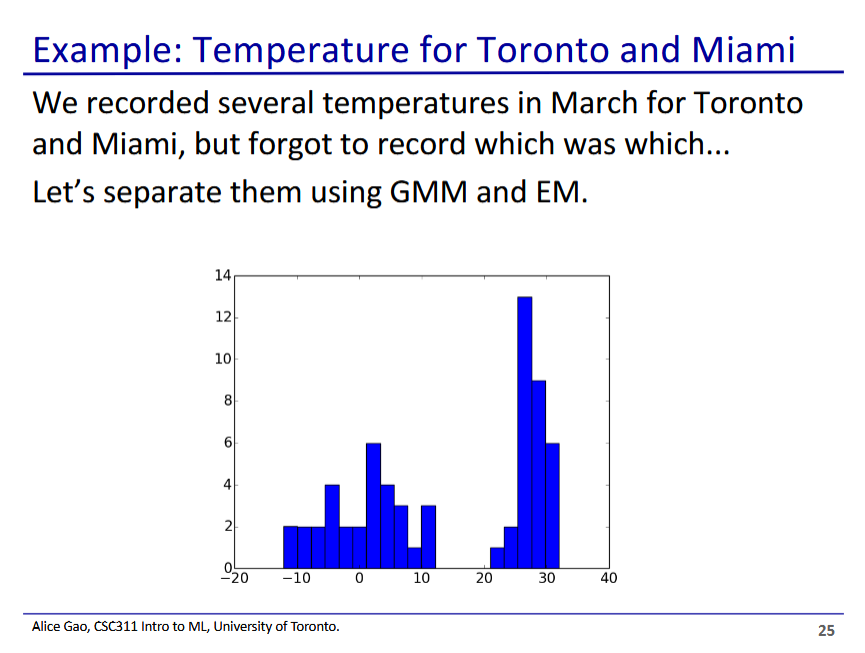
* We iterate between these 2 steps until convergence
* In the E step we are estimating the expected value of the latent variable ( is an estimation of )
* In the M step we update the parameters by maximising the likelihood of the data
  + Assuming the latent variables are observed (assuming our estimation is actually true)
* We can start the algorithm off by choosing random parameters for the clusters



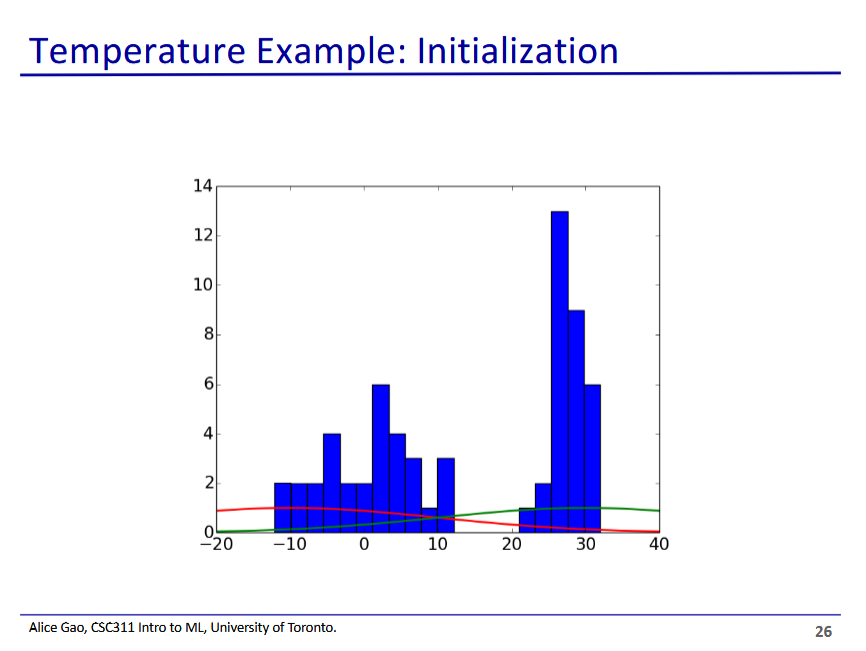
* We saw this equation in previous slides
* Denominator doesn’t matter, we include it in the normalising constant
* We try to find the cluster that has the largest



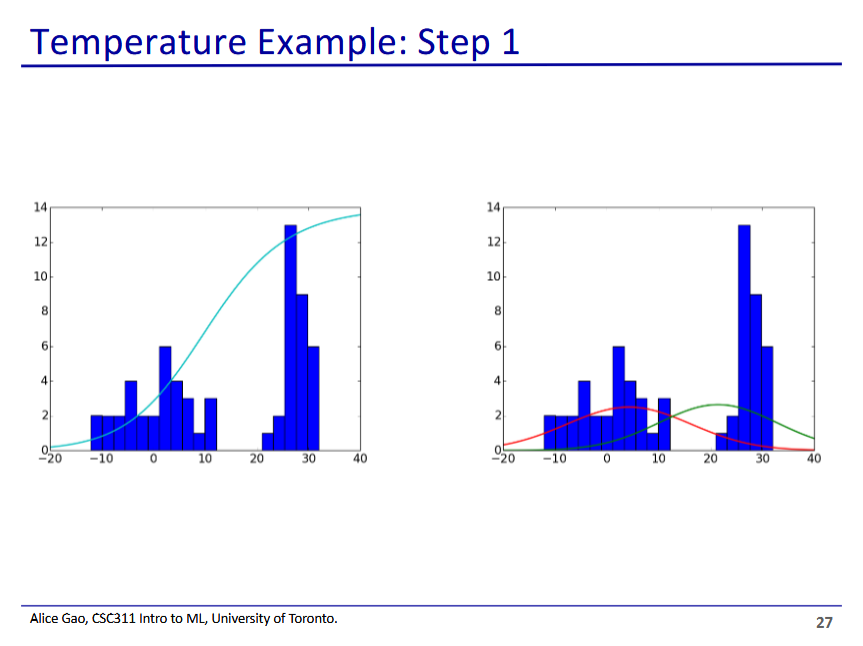
* Now using the assignments, we update our for every cluster
  + Ideal values are the empirical ones for points assigned to each cluster
  + These should look familiar



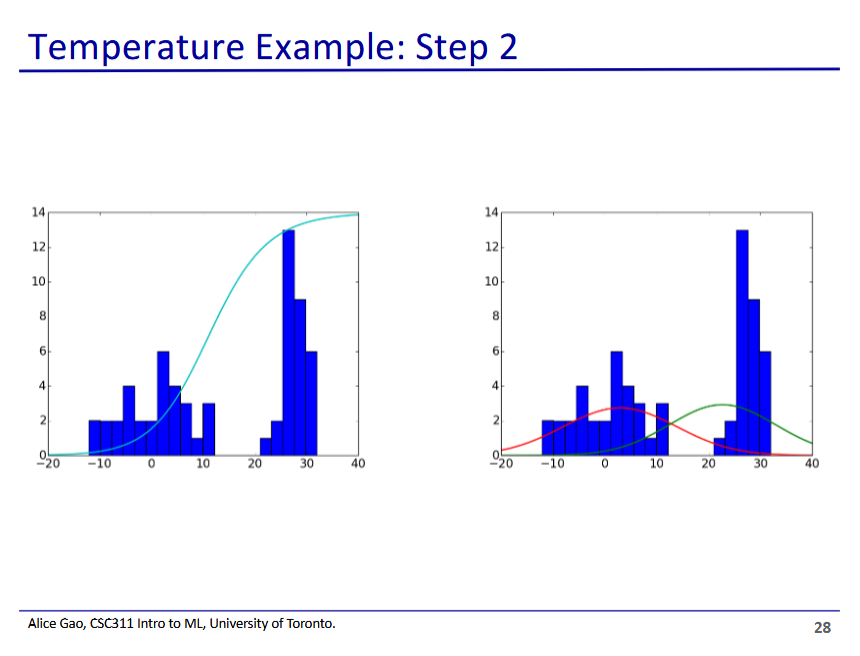
* We expect there to be 2 clusters, but we don’t know which temperatures belong to which
* It is obvious looking visually at the data, but we can do this using GMM



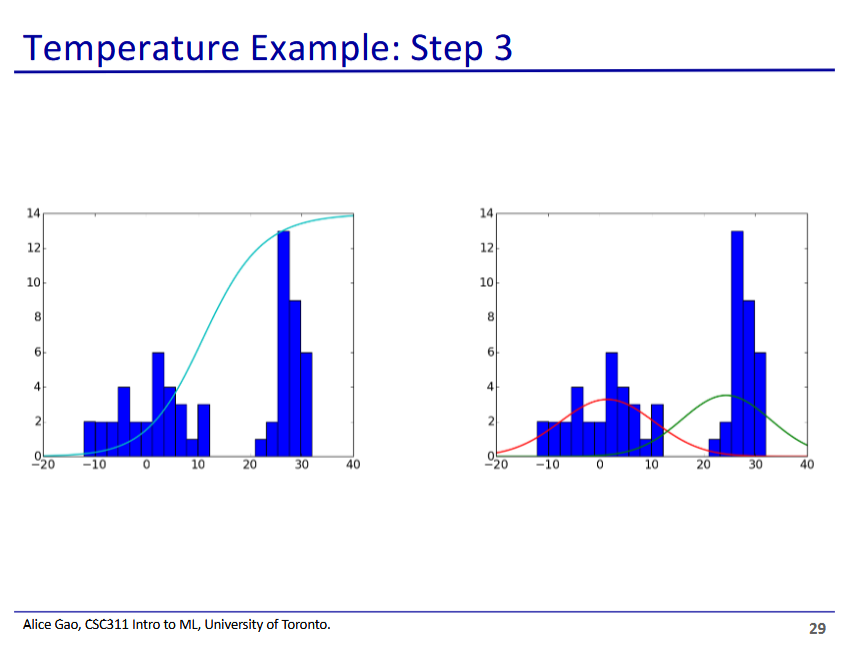
* Our distributions are 1D gaussian distributions
* We start by picking a random mean and covariance for each of them
* We also pick a mixing weight

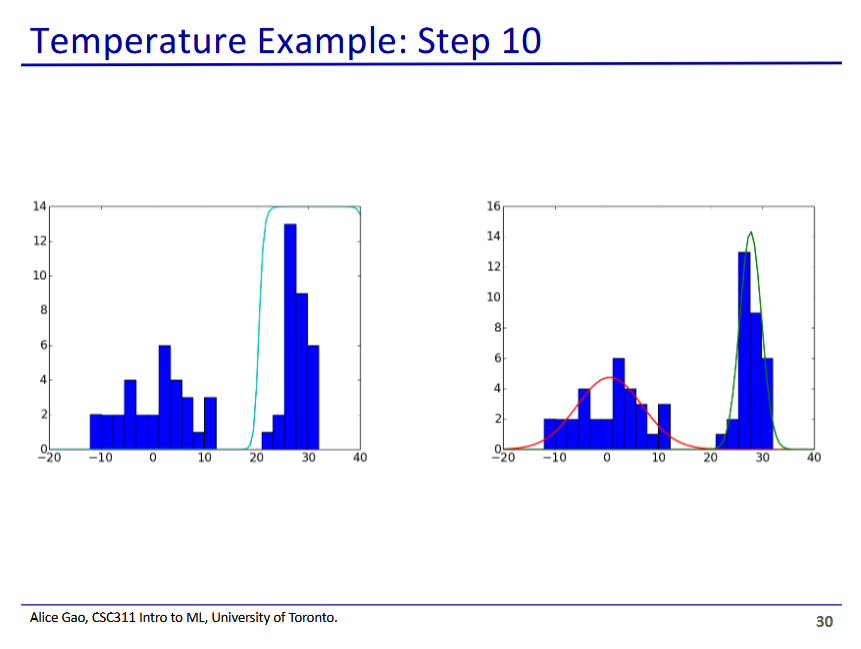


* Left is the E step, right is the M step
* In the left, the curve is the probability/likelihood that the data point with the given temperature falls within the green cluster

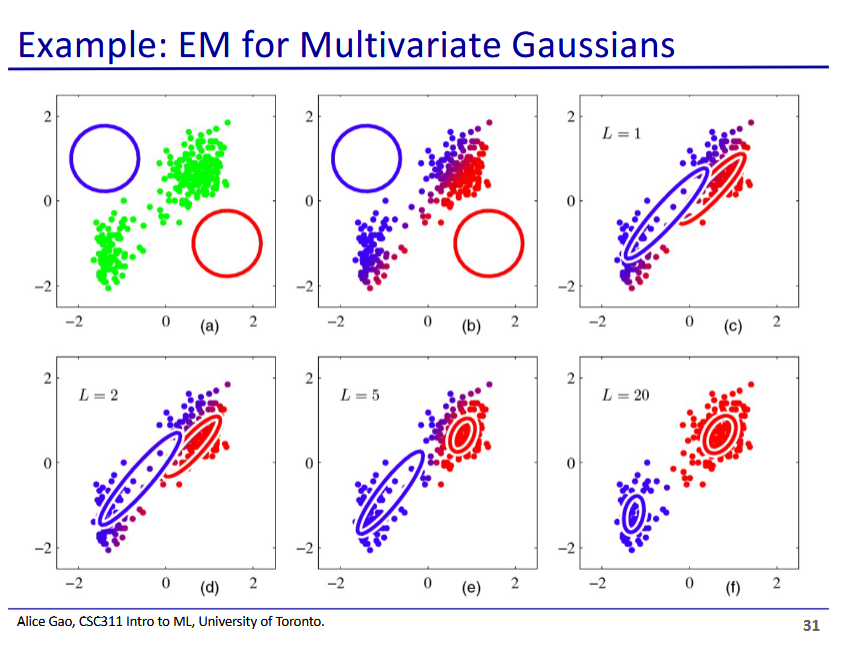


* Curves have shifted slightly
* E-step curve becomes steeper - we are becoming more confident in assigning points to clusters
* M-step curves becomes closer to the ideal Gaussian distribution

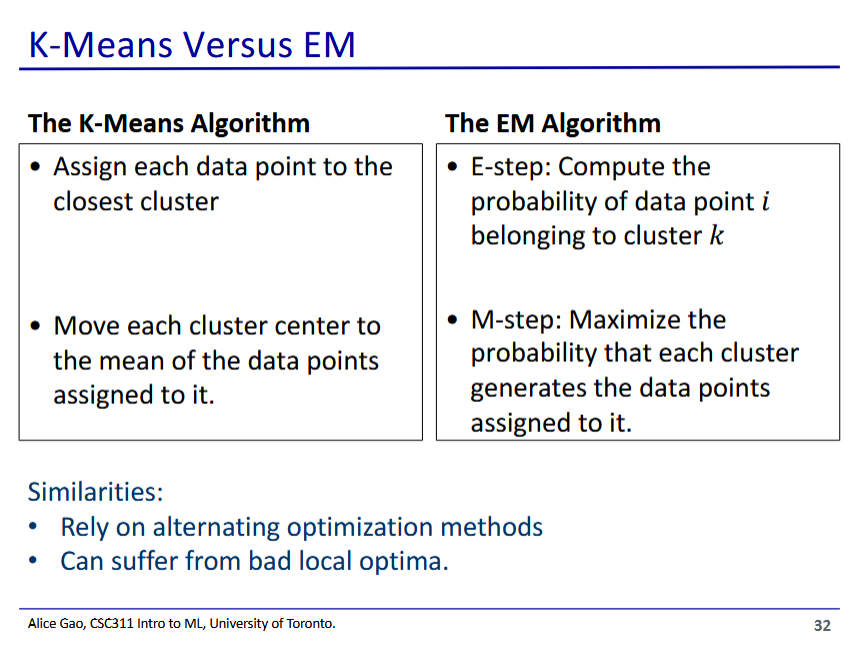




* Curves now almost completely fit the data



* Visual example for 2D gaussian
* Notice how the Gaussian clusters are not spherical



* EM and K-means have a very similar structure and problems

