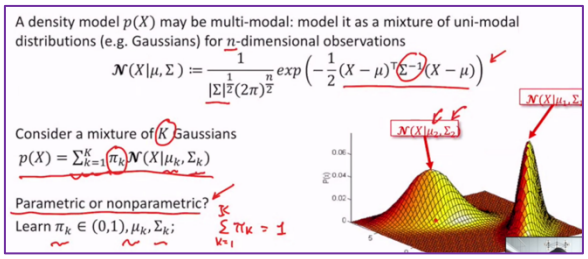
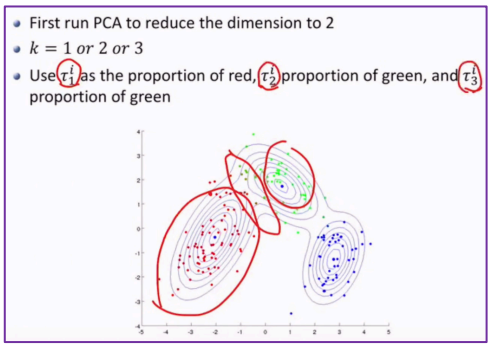
Today we will talk about Gaussian mixture model and EM algorithm.

**Gaussian mixture model (GMM)**: Last time when we talk about density estimation, we briefly mentioned the idea of mixtures of distributions. We have seen an example, remember the winded example where we can clearly see there are multiple components in the distribution of the data. it's quite natural to introducing mixture of distributions as an idea. One of the most commonly seen mixture models is called the Gaussian mixture model. And this is a mixture of a bunch of typically multivariate Gaussian distributions. Each one of them represents one modal in your distribution. And here this plot shows the mixture of two multivariate normal distribution Gaussian. And you can see each one of them can represent one node and that node has a particular center. Shape means the orientation, how flat it is, how round it is. Those are controlled by the covariance parameter; the center is controlled by the mean parameter. And I can also assign a weight to each of the Gaussian component that tells us the proportion of each of the Gaussian contributing to the overall distribution. in the end, that gave us the PDF, the probability density function of the Gaussian mixture model. A lot of times referred to as the GMM, Gaussian mixture model. suppose you have k Gaussians, and the Gaussian mixture distribution is the summation of the $\pi_k$ (represent the weights given to each of the Gaussian components), a mean vector $\mu_k$, and covariance matrix $\Sigma_k$; each Gaussian component can have a different mean and the covariance matrix. And here as a reminder, this is expression of a normal, what does it look like? So here, this denotes the determinant of the covariance matrix and n is the dimension of the data. with zoom we observe n-dimensional observations of data has n features. And here, this quadratic form, you can see involves the inverse of the covariance matrix. as you can see from this expression, a bunch of parameters involved in the Gaussian mixture model involving the $\pi$, these are the weights. The $\pi$ is between 0 and 1, and the $\pi$ has also to sum up to 1; sum of over all the k = 1 and the mean covariance parameters for each of the Gaussian component. here comes a question. Looking at the expression, do you think Gaussian mixture model is a parametric or non-parametric expression for density. We talk about this concept in the last lecture for density estimation. it's actually somewhere between parametric and non-parametric. It is parametric because we're sure a Watson parameter to be estimated. It is considered as non-parametric sometimes because Gaussian mixture model is actually quite flexible way to approximate the true distribution. Meaning that even if the two distributions is not exactly Gaussian mixture, something else, that using gas mixture, using the right number of Gaussian components, setting the right parameters, this can become a pretty good approximation to the true distribution. that's why we're studying Gaussian mixture model here because it's quite flexible, quite useful to describe the actual distribution for the data. And for example, this is the model very commonly used in image analysis. you can look at the distribution of bunch of images, it can pretty well be described by Gaussian mixture model. And we can also see some examples in your homework. For example, the handwritten digits. Very simple images.

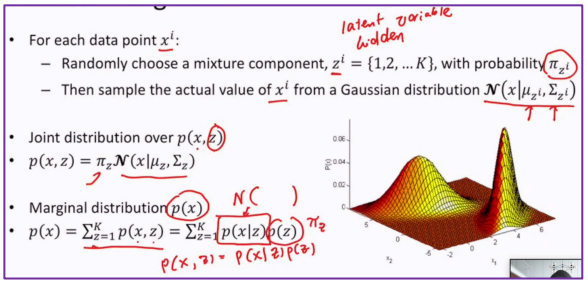



**Mixture of 3 Gaussians**: Let's start to see how we estimate Gaussian mixture model. as we mentioned, motivation is to study real data. And when you realize there could be multiple components, it's very natural for introducing Gaussian mixture model to model the distribution. We've seen an example from last lecture, the **wine data**, and when we perform the PCA to reduce dimension of data to 2 and plot a scatter plot of the principal components. And visually you can tell there seems to be three clusters. Three clusters, three modes in your distribution. And this also makes sense because if you look at the data, the data is collected of the wines. These are the measurements in the wines from three different places in Italy. it could be that these wines are produced in three different places. They have different ways of making wines, so the composition is different. It turns out that the distribution of the competition have three modes. if it wants to fit a Gaussian mixture model to this data, and we look at this you realize probably we can do three components of Gaussian to approximate distribution. The question is, how do we use Gaussian mixture to approximate data? And so here we have some notations. We explain the meaning of this notations later on when we start to introduce the mixture model, inference, and estimation. And it will start to make more sense. Roughly you can think about for each data point, we want to make k's does belong to the first, second, or third Gaussian component. And these variables $\tau$'s will tell you for the i data point, what is the probability or likelihood that ice data point come from each of the k's components. for example, this data point here, you can see, and the algorithm tells us, this data point is most likely to come from this first Gaussian component. That's why it's colored in red. And these data points are pretty much colored in blue. And you can see in the middle of these data points are not red or green, so it's a mixture of colors. these data points are harder to tell that belong to the first second Gaussian is roughly what tells us. And these variables are quite important. Parts of the algorithm we'll talk about next. I will explain where they come from.

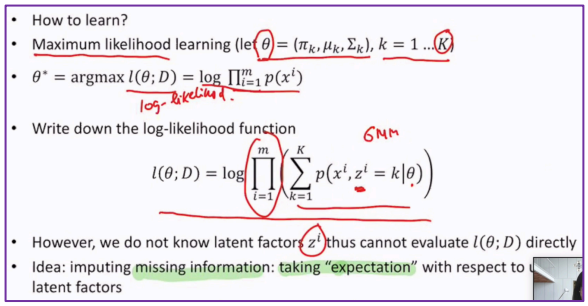



**Understanding GMM**: Let's understand the Gaussian mixture model better. when I write down the PDF of the Gaussian mixture, what does it mean? That means if I draw a sample $x^i$ from the Gaussian mixture model, there's something going on. First, I randomly choose one of the components, and probably I choose that component is according to $\pi$. this $z^i$ is a hidden variable or latent variable, we don't know, somebody pick one of the components for us and its probability $\pi_{zi}$ is proportional to their rates. once you pick the component, you're going to draw an actual sample $x^i$ from that particular Gaussian distribution, you're going to use a particular $\mu$ and $\Sigma$ to degenerate

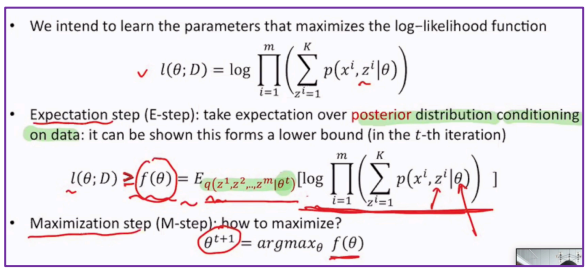
in random Gaussian sample, you can view that sample in Gaussian mixture is drawn from this two-stage procedure. if I look at the joint distribution of x and z, and essentially the joint distribution is going to be $\pi_z$ (the probability of the hidden variable) times N (the probability of generating actual x). And of course, I don't know the z, that's the beginning variable that I introduced here for us to understand the procedure. the marginal distribution in the end, the actual Gaussian mixture model is going to be this. we marginalize out the z that correspond to summing the joint distribution of x and z over all the possible values of z. intuitively what we can understand is that x can come from any of these hidden components, and then the total probability for my x is going to be summing over all of these z, possible hidden steps or hidden latent variable. And so, since I can write the joint distribution as x given z times the prior distribution marginals for z. I can write it this way. as you can see, if I decompose it this way, this becomes the multivariate normal distribution ones and tell you which component is sampled from then p(x given z) is going to be normal. And as p(z) is going to be determined by this $\pi_z$ factors. you can see now how Gaussian mixture model is a wide written and from essentially that implies there's a two-step sampling happens when you draw a sample from Gaussian's mixture.




**Learning Parameters**: The next question we want to address is how we learn the Gaussian mixture model. By learning a Gaussian mixture model, we want to be able to fit the parameters to data. the parameters in Gaussian mixture model are rates of each component, the mean vector, and the common matrix. we collect them together and represent using this data here. we have k sets of parameters. The k is the number of Gaussian components. of course, there's also the question, how do I decide k typically, before you fit Gaussian mixture model you would pick a k as a parameter to your model, which decides how many Gaussian components you want to fit into. And how to choose k, that's the question of model selection. You can use, for example, cross-validation or other metrics. We'll talk about this later when we talk about variable selection and model selection. To fit the model to data, naturally we will use maximum likelihood estimate. This is one of the default or commonly used methods today to estimate parameters of your model. And it's particularly useful when we have a probabilistic model like in this case Gaussian mixture, we can describe the distribution of data, it's where we actually use maximum likelihood. To write down the likelihood function and maximize the parameter with respect to the likelihood function. this is going to be our strategy here as well. I'm going to find the log-likelihood function. This is our log likelihood l(θ;D). I take the log of the distribution of my data in here, samples are iid following the Gaussian mixture. I'm going to write them as the product ∏of each individual sample's likelihood. That's it. when I tried to write down the log-likelihood function, this is going to be the expression, this is a product ∏, so multiplying this over all m samples. And inside this bracket, this is the Gaussian mixture model expression. Gaussian mixture model using this expression here, explained. Each p(x) can be written as the maximum addition of the joint distribution of x and z. And here is our parameter that we want to be able to solve for maximum likelihood. it seems to be okay, we already have a log-likelihood function, then the rest of the problem is to try to maximize this l. However, there is **one problem** standing out. You can see this likelihood function involves a latent factor z. This zi, it tells us which Gaussian component i sampled come from. And we don't know this latent factor, because we only observed the x, we only observe a bunch of data. imagine that you have at least a handwritten digit image, IDs are generated from 10 different possible digits, 0-9. However, we only observe a mixture of them, we don't know which digit it belongs to. we lose this latent factor information by only observing the x. because of that, even if we can write down the log, I hit function this way. We cannot directly maximize it because we have missing information. This zi is missing from our dataset. this is caused by the nature of the dataset. this essentially is a missing information. This missing information. how do we deal with that? There's a very natural idea when we have missing information is let's try to take expectation. Taking expectation means, given whatever we have, we only have x. Can we make a guess about the latent factors? Make a good guess, make the best guess possible about latent factory. This best guess idea is captured by taking expectation of zi with respect to given information which are these xi's. That's the most important idea in EM algorithm. We're going to describe next, which is, if we have missing information, if you have latent variables, we don't know the likelihood function, let's try to take expectation with respect to the unknown or missing information by conditioning on the information we know.

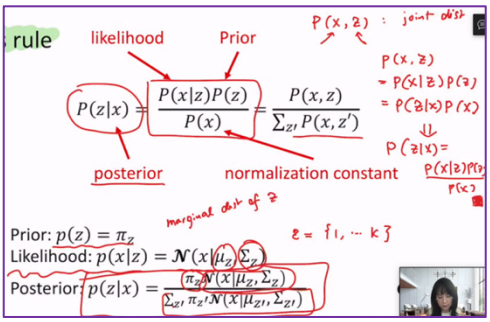



**Details of EM**: Now I'm going to start to describe the EM algorithm. EM stands for **Expectation Maximization**. Expectation maximization means EM algorithm. you can see it involves two steps: E-step & M-step. That's why it's called EM, iterating between the two steps. You first compute the E and then to the M, and then iterate E and M again. this is the iterative algorithm. And you can see the last iterative algorithm we see is k-means algorithm, iterative. Here is another example of iterative algorithm. we want to maximize this log-likelihood function. However, we cannot directly do it because we don't know the zi's. so, we want to be able to deal with that by taking the expectation of the log-likelihood function with respect to these (2nd eq) zi's. how do we find good expectation or what is a good probability density

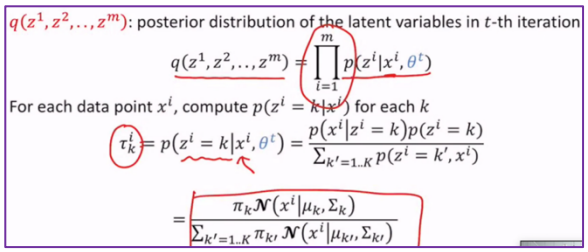


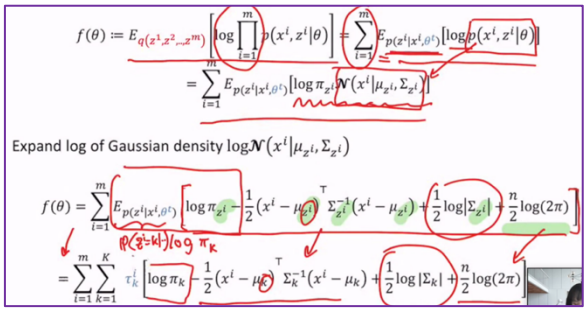
function, therefore, the z is actually taking expectation? here we're going to introduce in these q's. what are these q's? These q's are the posterior distributions of the z latent factors conditioned on the data, on the observation. And we're going to explain how to obtain this posterior distribution. there's Bayesian statistics idea involved here. That's another interesting and important concept here. in fact, we can show mathematically by taking expectation, this will give us a strict lower bound f(θ) or since lower bound last greater than or equal to, this lower bound will become a lower bound of our original log-likelihood function. that becomes a good idea. We can find this lower bound and maximize the lower bound in each iteration as a way to maximizing the rational objective function. once we find this lower-bound, which correspond to the expectation of the log-likelihood function conditioning on the observations and then we can find maximization of this lower bound in each condition and then this give us the θ. Why do we need iterative or recursive way to find the estimate? I'd use the following. when we do the expectation, which is respect to the posterior distribution. When we compute the posterior distribution, we still need to know or somehow have a guess about the value of the parameters. We need a model in order to find the posterior distribution. we don't know θ's. to find the posterior, we have to start from somewhere. Let's start with an initialization of the θ and using the initialization to find the posterior and then use that posterior to find the lower bound which is expected the log-likelihood. And this time you can see I still have the θ here, and that's going to be the argument then I'm going to maximize on the following equation. And once I find this, you have an update of the estimate for the parameters. I'm going to use this estimate to recalculate my posterior. in each iteration, the posterior is calculated using the parameters you have obtained or estimated from the previous iteration, and then you improve the estimate of these parameters by maximizing the lower bound. that's where the iteration comes from. We'll see examples that demonstrate this.

**Bayes rule:** I'd like to explain, what is this posterior distribution for latent factor here. to explain that we first have to talk about the Bayes rule. This is one of the most important concepts in probability and the basis for Bayesian statistics. It's basically a probability rule, but it's quite useful in analyzing data and study statistics. suppose you have two variables, x and z. You can imagine x is our observation, and z is some latent factor that's related to x in a certain way. we're going to describe the joint distribution P(x, z). once we have this and we're interested in finding out, there are a few things happening. there's something called the likelihood function p(x|z). it means that, conditioning means I tell you what z is, what should be distribution of x. And there's something called the prior distribution p(z). you can view this as the marginal distribution of z. given the marginal distribution of z which is the prior, and the likelihood, which is what z is, what should be distribution of x corresponding to that z? I want to figure out the reverse. What is the probability distribution of z given x? as if we're trying to make a guess, If I know x, what is our best guess about z? We're going to try to relate the posterior to the likelihood and the prior that we know from the model. This is what Bayes rule tells us. The posterior distribution p(z|x) is equal to the likelihood times the prior distribution divided by P(x). There's a very simple proof for this because P(x, z) be the joint distribution from the definition of conditional distribution is equal to P(x|z)P(z). If I condition on z and I can also write this in a different way, which is P(z|x)P(x). these two are equivalent. because of this, I can write p of z given x is the left-hand side, P(x|z)P(z) divided by P(x). so that's it. It's a very simple fact coming from the definition of conditional distribution. But this is a quite useful. Why? Because they you can see if you want to find the posterior distribution given the likelihood and prior, this is what you should do. You should multiply the likelihood and prior distribution and divide it by P(x). What is P(x)? P(x) is the marginal distribution, P(x) = sum of P(x|z)P(z) from z = 1 to k, if you only care about the distribution of x, the marginal distribution of P(x) is related to the joint distribution by summing over all the possible z, that's it. you can see this P(x) essentially is a normalizing constant. We call it normalizing because once you have this, then essentially this is ensuring that this whole thing will sum up to 1. that's Bayes rule, and once we have this, we can use it in other contexts in Gaussian mixture model to derive the posterior of the latent variable z condition on observation. in Gaussian mixture model, the prior for z is specified the weights of each of the Gaussian components as P(z). And the likelihood of the data conditioned on z, if I tell you which Gaussian component it is, P(x, z) is this multi-varied Gaussian with a corresponding mean and covariance matrix specified by z. z is this discrete variable, so it takes value from 1 to k, one of the k components. if I plug it in the Bayes rule, you can find easily that the posterior is given by this expression. this corresponds to prior distribution that's my likelihood and then you have to divide it by the marginal of x, that's basically the joint distribution, then sum over all the possibilities, that's it. that's the posterior of z given x. that's it.

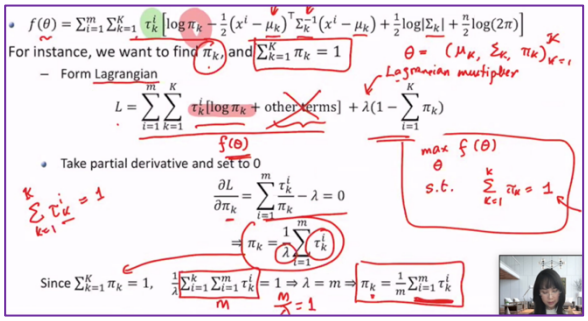



**E-step: Find the posterior distribution**: Once we have this, now our goal is to find the q, which is the posterior of all the latent factors in t-th iteration. remember that data is independent of each other and each of the z is the latent factor for each of the samples, so we can write the joint distribution of the posterior for the other of these: the product of each individual posterior distribution. we need to find out the $p(z^i, x^i)$, and here we're going to give it a notation, then we call this $\tau^i_k$. If you still remember a few slides back, I showed you a plot where I have this tie case. I give it a definition essentially when I plotted there was the posterior distribution of the i's datapoint belong to the case components. given the observation, you can see the only depend on the i's stamp because the samples are independent of each other. using the bayes rule, I can write this posterior in this expressive for expression.





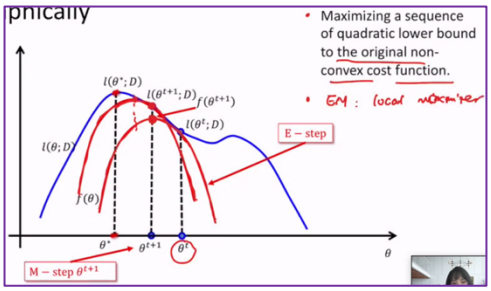

**E-step: Compute the expectation**: so, now we have the posterior, we can try to find this lower band, which is the expected the logic regression functions. That gives us a lower band, so the rest of two slides have some derivations but I want you to focus on the high-level idea instead of the mathematical details, but I want to show you whole thing for you to understand where things actually go but you can focus on the high-level idea. this is our expectation we're trying to find, using the property of log, I can write log of the product as a sum (log ab = log a + log b), then I can pull out the sum from the expectation because if patient is linear operator, so in the end, I have the summation over all samples, and I compute the log-likelihood for each individual sample with respect to the posterior, so I end up having this expression. when I'm writing down here, from here to here is I write the joint distribution of data and latent vector z as this so this is the π distribution of z and this is for the x. All right, so the next is once I have this question, I want to find the log of the Gaussian density, and so if you can refer back a few slides then you can know the expression for the multivariate Gaussian and taking log is given this expression (2nd equation). you can see everything is still within the expectation bracket. this is just taking log of the multivariate Gaussian density, and so from here to here, there's magic happening which is, I start to invoke the expectation. I'm going to try to take the expectations so there are a few things happening here. All right, so you can see this is actually what does it mean? It means that I want to find the expectation of z, this excitation she's like to z. And with respect to the posterior distribution of z, so you can see where z is. It shows up here and here, basically this is simply to play the role of selecting the Gaussian components, why it shows up in subsequent odd time; in the end I also have a constant here, which I don't really care because the constant is going to become still a constant. And so, if I want to take expectation of each of these, essentially, for example, this particular term is to say I can find log of $\pi_k$, and I had to find the probability for $p(z^i=k|x^i,\theta^i)$ condition. that's it. And this conditional probability of z given everything else is the quantity definition: $\tau^i_k$, that's the posterior of the i sample, the larger the case. from here to here, essentially this first term will become $\tau^i_k$ times log $\pi_k$. the rest of the terms are derived in very similar fashion. I can replace this with the k and find out what is the probability for $z^i = k$ and that's corresponding to two $\pi_k$. the second term is here, and this last term, the third term is here, and then this is a constant. this everybody is multiplying $\pi_k$, I can pull it out of the brackets so that becomes here. All right, so that's E-step.



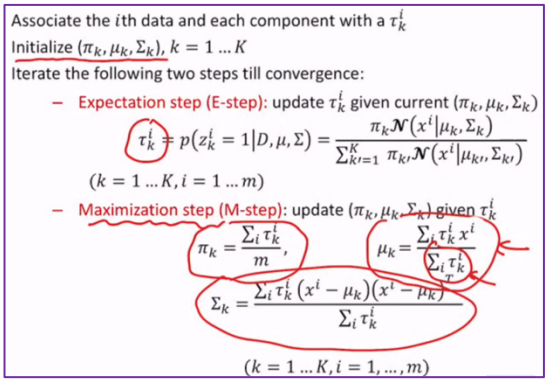

**M-step: Maximize f(θ)**: So, now we have this F function, which is the expectation of log-likelihood with respect to the posterior of non-living factors. I want to maximize f(θ),next step is a maximization problem. This is back to the parameters θ. The θ contains $\mu_k$, $\Sigma_k$, and $\pi_k$ for each k of the Gaussian components. you have K sets of parameters, that's our goal. And here, I'm going to illustrate just to show you how to find the weights, $\pi_k$. Okay, so let's take a look. this is a constant given to us. This is precomputed from the E-step, so we know what the $\tau^i_k$ is, so our optimum variables are this, $\pi_k$, and π only show up here. if I want to maximize respect to $\pi_k$ only one term matters to us, so I'm going to pull it out. And so, the other terms is involved, the μ, Σ and so on. I worry about them later when I tried to maximize this effect to μ and Σ. to find out the π that maximizes this half, I care about this first term and some other terms have declared in this step. And moreover, don't forget, I have a constraint that the sum of the π had to be equal to 1 because the meaning of the π is the prior distribution for the z and so themselves have to normalize to what. that's why intuitively it also starts the probability of the z being one of the components have to be, they probably have that so much from 1. It has to belong to one of the components. essentially, we're solving an optimization problem. We're trying to maximize this sub step max f(θ) with respect to θ, subject to constraint. This is the constraint optimization problem. in the next lecture we'll talk about this in much more detail, how to solve constraint optimization problem. For now, we have one equality constraint, sum of the $\pi_k$ have to equal to 1. you going to form the so-called Lagrangian function. The Lagrangian function consists of the original objective function, here this correspond to f(θ), and I'm going to introduce so-called Lagrangian multiplier λ. again, we're going to talk about solving optimization problem in much more detail in the next lecture for now I'm just illustrating how to do it. the Lagrangian multiplier, λ, and then here you can see I write my constraints as 1 – this (sum of π). this when a constraint is hold, then this should be 0. That's the idea. I have λ introduced, and then once you have that I have this out. Lagrangian function combines both the cost function and the constraints as what function. I want to solve the π using this Lagrangian function. The first step I do is I take the L, which is back to take the derivative of L with respect to $\pi_k$ and set it to 0, so that become this expression. from here I can solve the $\pi_k$ and in terms of the posteriors and λ. we don't know what λ should be, but now let's figure it out. You can use this constraint to that $\pi_k$ have to sum up to 1. I plug in an expression of the $\pi_k$ from here, and the next expression. you can see basically have this, 1 over λ, and inside I sum over all the posterior, over all the samples, and all the Gaussian components. Here, this should be k. And so, this should be equal to 1. in the end, I got λ is equal to m. this whole sum here is equal to m. Why? So, because first of all, the sum of $\tau^i_k$ over the k is equal to 1. why is that the case? Because this tells you that this means the probability of i's sample in the k's component. the i's will have to go onto one of the components. this have to sum up to 1, if you sum it over all the possible k. here, one of the sum is equal to 1 and you have m samples, you sum this (sum of (sum of $\tau^i_k = 1$) = 1*m) over m samples, so you end up having m, you have $m/\lambda = 1$, $\lambda = m$. from here I can find the final expression of my $\pi_k$ using

here. you can you see $\pi_k$ is summing all the posteriors over all the samples, and then divide by m. that's the average posterior for the k's component. It actually make a lot of sense, that's it for the $\pi$'s. And so similarly we can also find the posteriors for the $\mu$, and we can find the maximizer for the $\mu$ and the maximizer from $\Sigma$.

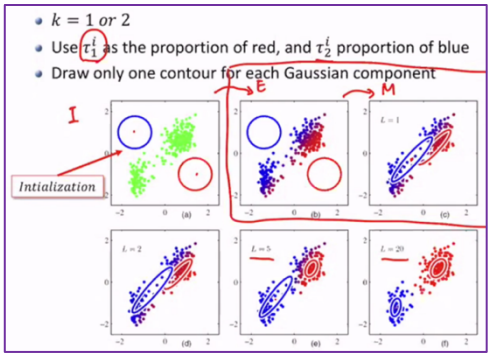
**EM graphically**: I'm going to summarize EM algorithm in next few slides. But first, intuitive was happening here, as you can see in each iteration of the EM algorithm, the E-step essentially is helping us to find lower bound to the original objective function by introducing this posterior invitation, essentially, if you look at it what we derive here, this f is a quadratic function. we find a quadratic lower bound locally to the original objective function. And in the M-step, we're maximizing this lower-bound to find the next data points. here this illustrates the idea is starting from $\theta^t$. This is the current value of the parameter and others place we find this lower bound which is the local lower bound to the original objective function, and then we maximize it, and we end up in this point, this is the actual objective function of log likelihood, and you can see this indeed is a lower bound, and so you get to this point, try to find another lower bound approximation to the original objective function and that having this function here. And then you try to solve it again and then maybe end up in here. after a few iterations, this can converge to one of the local maximizers. And that's what the EM algorithm is going to terminate and return to you. EM algorithm will find a sequence of quadratic lower bound, the original non-convex cost function. the cost function integrational problem we're trying to solve is actually quite non-convex. that's the first thing. And the second is actually the EM because of the nature, you can see it only find a local maximizer. EM only find a local maximizer.




**EM algorithm**: here's a summary of the EM algorithm and summarizing what we have derived so far. we will initialize a set of parameter values, the weights, the mean, and covariance matrices and integrate between the two steps. in each step, we find the posterior $\tau^i_k$ of the i sample belong to the k's component. by finding the posterior, we can use the parameter values in the current iteration. that means if you initialize it in the first iteration, we're going to use the parameter value from iteration to evaluate the posterior. Once we have the $\tau^i_k$, we can find an update of the parameters. in fact, the maximization step, find the parameter that maximizes the lower bound and its maximizing, all have closed some expressions. which making it quite simple, and we divide what is the $\pi$, that's going to be the average of the posterior over all samples for each component. in fact, you can derive, not super difficult. You can derive that mean and covariance matrix for the Gaussian component also takes pretty simple form. the mean for each Gaussian component is a weighted average of the samples. you can see it's quite interesting because the weights in this case correspond to the posterior of i sample belong to the k's component. to see that the pattern here, you're trying to assign the i sample to the k's component that your assignment is in terms of this posterior probability $\tau^i_k$. you going to normalize it divided by, if count, what is the total percentage of data points assigned to the k's component? That's roughly tells you the percentage of data points. actually, this summation here doesn't need to be between 0 and 1, because we talked about last time that $\tau^i_k$ if sum over all the k does have to be equal to 1 for any of the data point. But here this summation is respect to the data point. this summation doesn't have to be less than 1. Similarly, the covariance matrix, you can see it looks almost like a sample covariance matrix. Except for that, you have these weights assigned to each of the data points according to the posteriors. All right, so that's the EM algorithm.

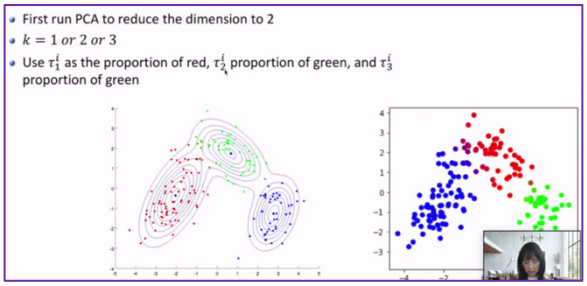



**Expectation-Maximization Iterations**: so, what happens in the EM algorithm, here's an **illustration** of the steps and they are very simple, two-component EM and here for example, we assume two Gaussian components, initialize two Gaussians, two bases to have the centers here and here. choose a covariance matrix to the identity matrix that corresponds to specify two circles, then we're going to illustrate the $\tau$'s using colors; $\tau^i_1$ & $\tau^i_2$ as a proportion of red & blue. that you can visually tell us the assignment of the data points. this is neutralization and after one iteration, where you calculate the $\tau$'s using initialize the parameters. You can see ones that are close to this Gaussian component are pretty blue and the points that are close to this component are pretty red. that's quite intuitive. And the data points in the middle have a purple color. That's a mixture of blue and red. And those points in the middle are indeed harder to design and to be that represent you're always in also knows that to eat that the data points in the middle, very last certain which side? Because I need to base on the current Danish radiation. This is E-step, and from E to the next step, you calculate the M-step to update the parameters. you're going to recalculate the mean, the covariance, and the weight. you can see this is a very drastic artists. The centers of each Gaussian components are dragged towards the middle. And the new center and the shape of the Gaussian components also becomes different. And you can see the data are spread in this direction. And then the covariance matrix

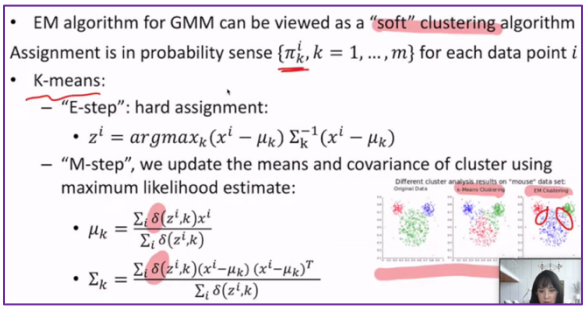


after one update for both components seems to be stretched along this direction to capture that the variance and all that direction is pretty large. quite interesting. From here to here. This is counted as one iteration, one EM. And then in the next iteration, this is after another EM. And then you do a reassignment and update the parameters. And after five iterations, after 20 iterations, you can see as you're getting more and more iterations, the data are re-designed, meaning, I call it a zine. Essentially going to calculate this posterior is, we'll calculate posteriors and that changes the color of the data points. And then you recalculate and update the parameters with the center and the μ and Σ which described the shape. You see both gradually converge to the correct center and crunch shapes. in the end in this case, we get the correct results, the desired result. Each of the Gaussian components is capturing one of the data clusters and then the center since we centered at each of the data cluster. And the shape of the coverage metric is capturing the current shape. this is a pretty good case where we actually convert pretty nicely to a local minimum, excuse me. Local maximum actually gives you the desired result.

**Demo: Wine data**: Next, I would like to show a demo how this works on real data. And this is the wine data we have seen last time in density and estimation. Here we're going to use a different approach to find density to fit a Gaussian mixture model weight. here I would like to show you this is the Python demo. you can see console is reading the data. then as a preprocessing step would do PCA. And to project the high-dimensional data into two-dimensional, so projected into two principal components. Now we going to fit three Gaussian components. then here's initialization. We initialize the π and the mean and covariance are randomly initialized. And then from here, we can start to run the E and M algorithm. here that's the actual steps happening. You can see this correspond to the E-step to find the $\tau^i$ case correspond to the posterior for each component, for each data points. this is the M-step. Once you have the τ, you can update the π, mean and covariance matrix. I'm going to show the density. What does the Gaussian density look like and examine and so on in this demo. I going to start to run, and you can see it start to run very quickly. And here you can see the colors of the data points are changing gradually it's designed into three components, data points here. And this designed to green, red, and blue. Algorithm has converged after 57 iterations. It seems we're getting a pretty nice result; we have three components separating the three cluster of data. Let's try it again. And because random initializations, so the number of iterations it takes to converge may be different. This might actually a lot faster after 18 iteration coverage. Coverage different local minimum, but those sensitive a desired result. And here I also want to show you another demo doing the same thing and you read the data, do the PCA, two-dimensions. And then here is evaluating a grid point so we can actually visualize intensities. And so here is to initialize the parameters for π, μ, and Σ. And then there's the actual yen iteration. The E-step to find the τ, the M step to find the mean, the update for the π's. And then update for the mean and the. let's see, this is also shows the contour of the Gaussian for the three cost components. And the color data shows you the $\tau^i{}_k$. This might actually converge a little bit slower. And after 100, this is what the coverage do. that's now a demo here. so, we have seen how this actually performs on the wine data which seems to extract the desired results.

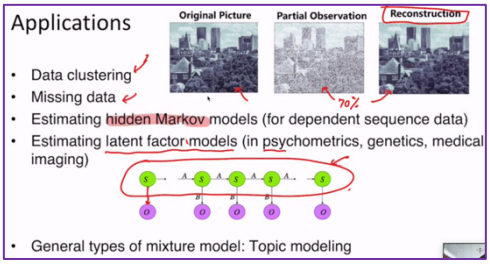



**EM vs. K-means**: At this point you might have questions. what we're doing here seems to be quite related to what you may have seen before. We're finding clusters for the data points. And the clusters are described by one Gaussian component. we have seen the clustering algorithm before. What is that? The k-means algorithm is also to find a cluster. natural question to ask is, what is the connection between the two? EM algorithm for Gaussian mixture can be viewed as soft clustering algorithms, you can see what happens in each iteration we're calculating is posterior. If the posterior tells you, given the observation, what is the probability of the highest data point belonging to the case Gaussian component? you can view this posterior probability as a probabilistic design end of the highest data point to the case component. instead, in K-means the assignment in datapoint is the hardest. In each iteration, you will assign 100%, one particular data point to the closest centroid. EM is not 100% designing in terms of probability. that means you have a percentage, is percentage represent your answer. that's why we see these demos that the color of the data for the data points that are somehow on the boundary between two clusters, the colors are not very clear. that's represent the belief of uncertainty exist for these data points, but not 100% sure which component you should want to. the self-assessment is one of the reasons sometimes EM can perform better than k-means. here are three examples to show you the performance of K-means versus EM, you have three clusters; two are small, one is bigger. if you run K-means clustering, you can see data points here that actually belong to the bigger ring cluster are mistaken for as the two small clusters. for the points on boundaries of the clusters, the K-means because it has a hard assignment, and in iterations, my 100% design is data points to one of the clusters. it's much more difficult to recover from the state vs EM always seems to magically do a pretty good job separating the three clusters. EM is more flexible in the sense because you only assign a fraction of the data points to one of the clusters, so that seems to be doing better for the data points on the boundaries. illustrating the connection between the two, mutation is, you can see if we have a delta, this δ is hard assignment. The δ is going to be either 0 or 1 so if the highest data point is assigned to the k, then this δ is 1 otherwise it's 0. If you have that hardest assignment, then we should cover the clustering algorithm, that's the correspondence versus you can see in EM is δ is actually applied usually by the π, which are self-designing. All right, so that's all we want to talk about EM for Gaussian mixtures.

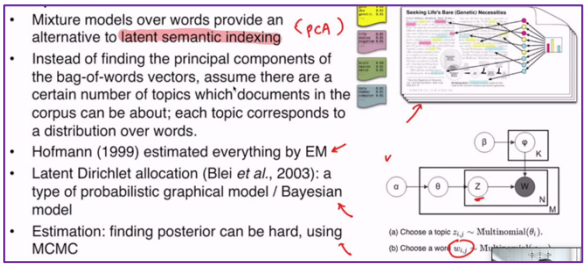


**History**: The original idea of EM was introduced in this paper in 1977 by Dempster, Laird, Rubin. The convergence of EM to local solution was to provide in chat by Jeff Wu in the 80s, this is very useful. we talk about one example of EM for solid Gaussian mixture where the misinformation is because we don't know latent factor. But in general, EM can be applied to many problems where you have missed information or related factors. This could be due to data you don't observe or latent factors you don't know. EM can be derived using a similar principle, finding the posterior, taking citation of the log-likelihood with respect to the posterior and maximizing the expected log likelihood. This general principle can be applied to other problems as well. In general, there's no guarantee to converge to a maximum likelihood estimator. Sometimes it works pretty well.

**Applications**: Here's some additional applications. We talk about one, you can see fitting a Gaussian mixture model using EM and, in the end, this can be used for data clustering and can achieve better performance sometimes than K-means. It can be a little more expensive as an additional static. These are posterior distributions but sometimes gives you better performance. it can also be used for inferring missing data, here is one example. If suppose this is original image, this is the picture removing 70% of the observation factors through a lot of pixels. can we recover the picture by filling out the missing entries, imputing missing entries. we want to recover this to good procedures. This is the reconstructed image. EM is also quite useful for estimating hidden Markov models. This is a model for sequence data. One year sequence observation have dependence, current one depend on the past. … for example, speech analysis, for modeling sequences in DNA, genetic data, useful for modeling human activities. these are all many places hidden Markov model has been used.




**Topic Models**: Here's a example of a hidden Markov model. The latent states are dependent on each other. We can't observe hidden state, but we can observe the observations. In order to estimate the hidden Markov model, we have to be able to deal with an observable latent factor, so that's where EM algorithm can be used to write that. Estimating latent factor models is useful in psychometrics, genetics, medical imaging, text modeling. For example, the topic modeling is developed based on some of the ideas. Topic model can be used to model text documents and so this is an alternative way to model texts than latent semantic indexing. we'll talk about this when we talk about PCA. If you have a bag-of-words model, then you do a principal analysis, you have latent semantic indexing. Here, let's talk about looking at constructive topics out of your documents, the idea is based on imagining you have a text document, then look at the words appear in this document, and the frequency of the words in the document. The frequency of the words can be related to the topic is talking about. Here for example, this paper is about seeking lives genetic necessities, and the topics involves the gene and life and possibly computer as well. This topic, we don't necessarily know but because the paper is about this topic, so you can see a mixture of these words related to these topics. That's the idea here, view the documents as mixture of keywords. And if it depend on a few topics then the mixture of the keywords will depend on each topic, what is the distribution of the keywords. Here is an illustration of what a typical topic model looks like. For example, the Z representing topic in choosing one of the topics from a multinomial distribution and then this determines the underlying model so you can think about the model we talked about. You pick one of the Gaussian components here, pick one of the topics and then under that particular topic, you have a certain mixture of words corresponding to one. And one of the Gaussian components, you will draw a particular observation. That's most important idea. Of course, in any real applications the topic model will become much more complex, so you are introducing additional distributions on the parameters, so you have hyper parameters. in the end it becomes a more hierarchical Bayesian model, you are introducing distributions through parameters themselves. This is to make the model itself to become more powerful, more expressive, and so because of that, fitting the model itself can become more difficult as well. The early version of a topic model estimation can be done using EM. And so, when we are introducing more complex Bayesian model, and this can also be viewed as a premise of graphical model basic variables depend on each other in a graphical way and finding the posterior can be hard. We can use some more advanced techniques such as sampling technique, Markov Chain, Monte Carlo, to sample from complex posterior.




All right. that's it for today. Hopefully I have introduced to you one example of mixture model, which is Gaussian mixture model and how to address the model fitting problem for Gaussian mixture using EM. As you can see, the purpose is to introducing EM as well because it's a general strategy for dealing with missing data and observed latent factors, and so on. The principles are quite similar for these types of data. You try to find the posterior distribution of the unobservable condition on the observables and then find the expected log likelihood with respect to these posterior distributions. Then you' try to solve your model fitting problem based on that by maximizing prime years, for example. In the end, I extend this by talking briefly other applications that are related to the mixture models and using EM for making inference. All right, that's it for today. Thank you.