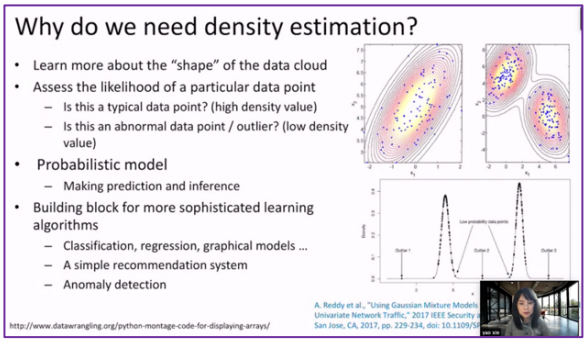
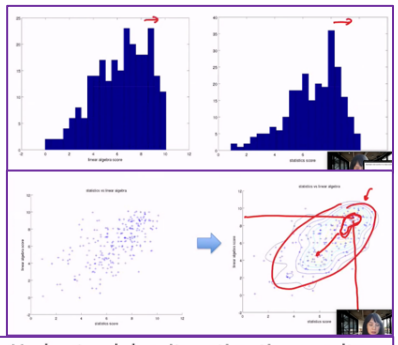
Today we will talk about density estimation, and we'll throw a little bit statistical flavor into our analysis.

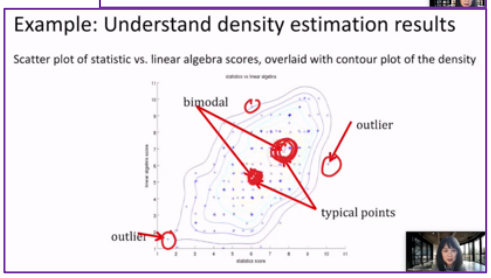
First question, why do we need density estimation? You can imagine this is another way to understand the shape of the data clouds. So far, we have talked about how to find clusters using k-means spectral clustering, and how to explore the intrinsic structure of the data using PCA and non-linear dimensionality reduction to find manifold structures. These are all different ways to understand the shape of the data clouds. This estimation can be another way to understand how data are distributed into space. This is useful in particular when we talk about how to build a model that interpret the likelihood for a certain data point to occur. It's particularly useful, for example, anomaly detection. We have training data and we fit a probabilistic model to that data, and that tells us whether or not a test point is likely to occur and your particular model, so it's quite important to get a density estimation. Bottom line, this is a building block that's very useful, fundamental to build more probabilistic learning algorithms.



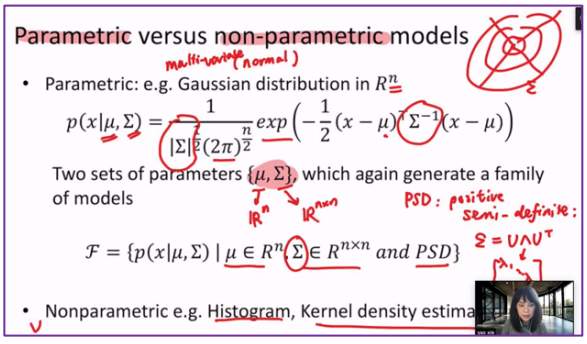
To start with, let's take a look at a very simple example. Suppose we have a task in the class. We didn't have a task in the class but just imagining there is one, and we collect the scores from the students, we have two subjects in a test : statistics and linear algebra. This is the result suppose and clearly this is what the professors really care about; to look at the distribution of the scores. Each data point is a pair of scores, received by the students. If we look at this distribution, what observation or conclusion can you draw from it. First thing, I would say, you notice that these two variables seems to be quite correlated. Why? In general, you could see people getting higher stats score will get a higher score in linear algebra. Not surprising because both are mathematics subjects. There are also outliers. In this sense, people who are really good at linear algebra, but maybe they haven't got a feeling about statistics, so their scores are not very high. This is particular outlier here. How do we understand more from this data instead of just looking at the scatter plot.

We can have the histogram. We have a histogram for each individual score, this is distribution for linear algebra score and distribution for statistics score, and we observe that there seems to be peaks that leaning towards the higher end. Most people seems did well in this test, and another observation is, you can see this noisy. There's randomness and fluctuation in these data points. If we want to generalize this to 2-dimensional setting, how do we characterize the distribution of the scores in this 2-D space, statistic versus linear algebra scores. This is a kind of plot we would like to have. This shows you a contoured density that tells you, for example inside here, that is most dense area, most of the students are in this spectrum. In this end, they have good stats and good linear algebra scores here. Outside of this peak, you can imagine it is like the height of this 2-dimensional distribution, and then it gradually decays over this direction, and it has this orientation like an ellipse in this 45-degree angle. How do we come up with this contour plot seems to give us a lot of information that is a subject of study, simply put in density estimation. The closer this provides more information and looking at this kind of density estimation, this will provide a lot more information than just looking at the scatter plot of the data. We can observe there seems to be actually bimodal. Now just the one mode, and this is the most likely outcome. You got a high stat in light linear algebra, but there's another mode here. You're somewhere in the middle of both score, and there are outliers. Those are points lining on the outskirt of the density function. Statistically speaking, or in talking about likelihood, that means these points are less likely to occur, these are unusual patterns, and these are outliers. The points in the middle are the typical points.

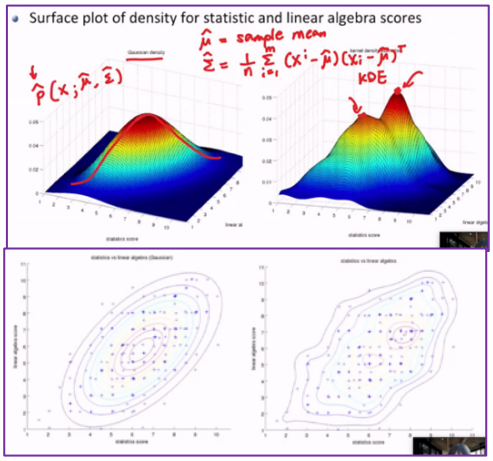




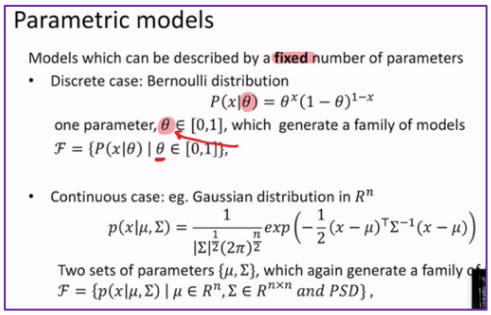
What are some typical ways to describe densities where in particularly probabilistic distributions. There are largely two families of method, **parametric vs non-parametric methods**. For example, in parametric method, the Gaussian distribution, is very commonly used. In this case, we're looking at a vector observation, so this is actually multivariate normal distribution with mean vector mu and covariance matrix sigma. If you look closer, this is so important I would say. A lot of students actually memorize this expression, and you can see it depend on a few things. Here, this is the determinant $\sum$ of the covariance matrix. This is the square root of $\sum$, and it is constant 2π and raise the power n/2. N is the dimension of data, and here you have this exponent, and this exponent gives you this Gaussian-shaped decay. The data goes to distribution centered on μ, and the sigma $\sum$ commerce matrix describes the shape of the distribution (how large of the variability across



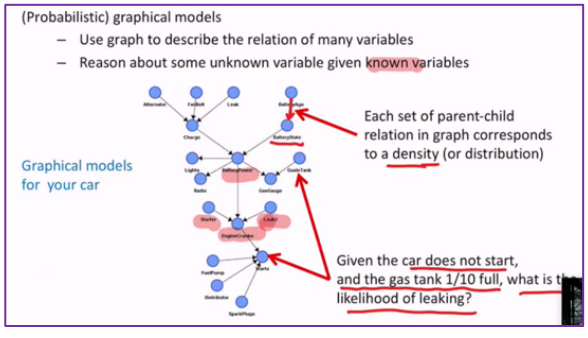
different dimensions, roughly speaking). The Gaussian distribution have two sets of parameters: mean vector (n-D) and covariance matrices (n-by-n D). Moreover, this covariance matrix for it to be a legitimate co-occurrence matrix, it has to be positive, semi-definite, PSD. What does it mean? If you look at the eigen decomposition of your $\sum$, all the eigenvalues for the $\sum$ are all non-negative for any of λs. The Gaussian distribution is completely parameterized by the mean and the covariance matrix. When it comes to density estimation, the problem becomes given typically the training data and try to estimate these two sets of parameters determined the mean and the covariance matrix. Besides this there's also very popular non-parametric methods. This kind of method doesn't assume any specific form for the distribution unlike the parametric methods. In parametric method, essentially, we describe the distribution by choosing a certain form of the function that's controlled by the parameter. In non-parametric setting, we don't pre-specify the shape of the distribution. Two most commonly used non-parametric density estimation are the histogram and the kernel density estimation (KDE).



Let's compare the two data; statistics vs linear algebra. If we fit the Gaussian density to this, essentially, we're going to fit a 2-D multivariate Gaussian, find out the mean and the covariance matrix. In fact, it can show that typically the $\mu\hat{}$ is estimated as the sample mean, the scores for these two subjects. That gives you the estimate of the mean vector. The estimate for the covariance matrix $\sum\hat{}$ is typically performed this way; $1/n \sum(x^i - \mu\hat{})(xi - \mu\hat{})^T$. That's a typical less estimating the covariance matrix. This left plot is the probability density function PDF when I plug in the estimated mean and the commerce matrix, $P\hat{}(x^i, \mu\hat{}, \sum\hat{})$. I'm going to put a hat on the top to indicate this is actually an estimate from data. What you can see, is pretty smooth surface. Parametric probabilistic density estimation typically gave you a pretty smooth result versus the non-parametric approach. This right plot is non-parametric approach based on KDE. I'm going to explain the details how this is done. But you can see compared to the result based on Gaussian density, this seems to be not smooth, roughly have some shape like noisy bumps here and there. But somehow, importantly, KDE captures the fact that you have two modes in the distributions. You have two peaks in the distribution, the two most common seen combinations of the scores versus, this is entirely missed from the Gaussian distribution density. Now you can see the trade-off between the two seems to be emerging out of the water, the parametric density estimation tend to be smoother, looks nicer, less noisy. But if you have a model mismatch, for example in here, you can lose some information by assuming a pretty strong assumption on the distribution. Versus KDE, this is minimum assumption on the distribution itself. The good thing is it's pretty flexible, but the drawback is it tend to be noisier and sometimes if the noise is too large, then you can start to lose track of the picture as well. How do we quantify these trade-offs is another interesting question I want to address here. This is another picture looking at the contour plots of the Gaussian distribution versus the non-parametric density estimation from KDE. And again, you can see this pattern. The Gaussian only has one mode versus KDE has two modes, but it's noisier and you all seeing is less noisy.

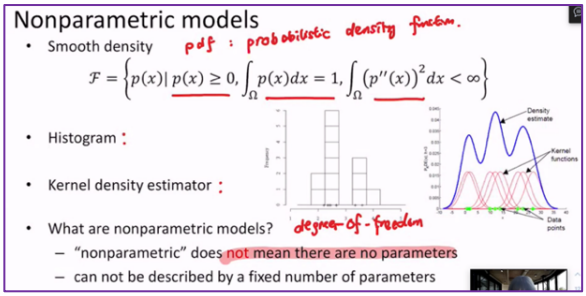


Next, I would like to illustrate a few examples of parametric models for the density functions and show how do we typically estimate and fit this model, meaning estimate the value of the parameters from data and how do we fit this model to the data using maximum likelihood. Parametric model typically can be described by a **fixed** number of parameters. For example, you can have a discrete distribution. Consider flipping of a coin. The outcome is binary, a head or a tail. To describe the probability for the chance of seeing a head vs a tail, we can introduce the Bernoulli distribution parameterized by the theta θ which is a parameter with value 0 or 1 and has a meaning essentially that is the probability of seeing a head. If θ is 1/2, you have a fair coin and have equal chance of seeing head vs tail. But if θ is not 1/2, you have a biased coin and have more chance of seeing one specific face. This parametric model will generate a family of models that is controlled by θ. Once we have this discrete model, for example, you want to find out based on your data, which is a sequence of coin tosses to try to estimate what should be the value of θ and that gives you the distribution function. The continuous case, for example, multivariate Gaussian gives you a family of a density function that is parametrized by μ and $\sum$.

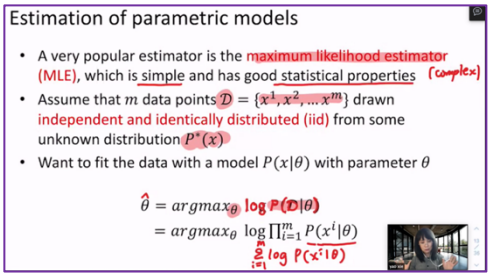


Next example is probabilistic graphical models. This uses probability to describe especially relationship between many variables. I would say this is becoming very popular in modern machine learning especially when you have complex data, you have lots of variables and you want to understand the dependence of these variables. If you have lots of data and treat these variables as random, you will consider using a probabilistic graphical model. This is a pretty versatile and flexible because in principle, you can use this to model any kinds of dependence between the random variables. Once you have this model, it is also quite useful to give some variables, to reasoning out about unknown variables, for example, thin up missing data or making a guess, making an

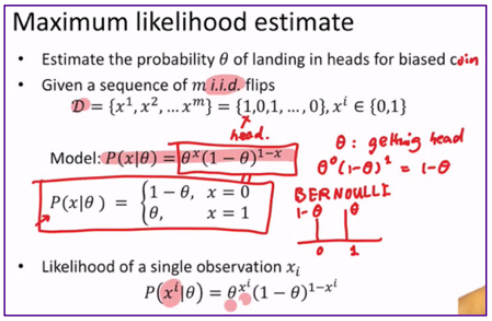
inference about another variable that depends on your observations. Here's one example of a graphical model for your car. Your car's status depend on a lot of variables, for example, BatteryState, BatteryPower, EngineCranks, Starts etc. The status of a car is running okey depending on all of these variables. If you mess up with the battery, then it's going to influence other variables. In this model, the variables dependence is described by a graph, typically a directed graph. Because the dependence can be 1-directional, 1 determines another variable. Each side of parent-child relationship correspond to a density. You would say, what is the probability of getting a certain battery state given its age? You want to model the dependence of battery state and age. Each of the arch corresponding to 1 particular probability density function and putting them together, you can build a complex joint distribution function and specify the whole model. For example, you can make an inference about given the car does not start and gas tank is 1/10 full, what is the likelihood of leaking? You can use the observations to try to figure out the status of other nodes.



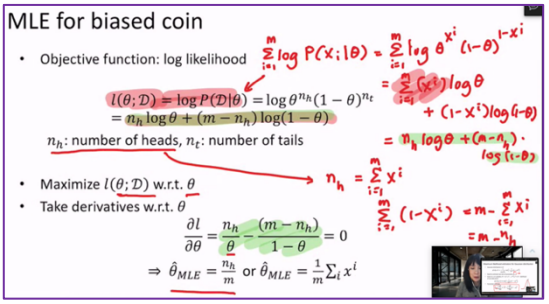
Here's the most general definition of Non-parametric model. It's basically any smooth density that you can think of. This is basically probabilistic density functions (PDF). A legitimate PDF should satisfy their non-negative, and it can be 0, meaning that, that particular act has 0 probability to occur. They have to integrate to 1, the total probability for all the space have to sum up to 1. Typically, when make assumption that have finite 2-order, their 2-order derivative of square have to integrate to a finite number. This is basic thing chose some irregularity word, the shape of the PDF. They cannot oscillate too wildly. Some typical way of getting p(x) is including histogram and KDE. What do we mean by non-parametric? Non-parametric model actually doesn't mean they don't have parameters entirely. What that really means is we don't describe the data using a fixed number of parameters. For example, in multivariate normal Gaussian distribution we assume the mean vector is a fixed dimension. The covariance matrix is parameterized by a certain number of parameters. But in non-parametric model, you will see that we don't want to fix number of parameters, understands the **degrees of freedom**. For these bottles are not fixed, the models themselves are quite flexible. We have talked about two kinds of models.



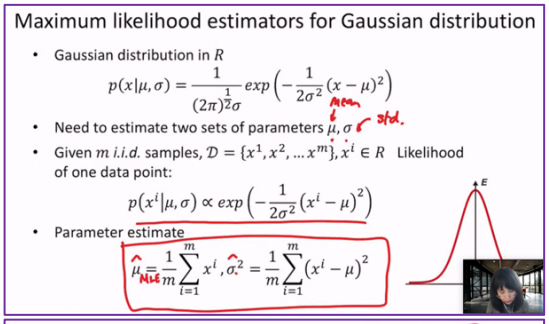
Next is to talk about how to estimate parametric model. A very popular way to get the estimation in machine-learning is **maximum likelihood estimator**, MLE, because it has good statistical properties that has been proven, and typically relatively simple to derive. It's a very general framework to get a reasonable estimate. Sometimes getting MLE for complicated probabilistic models can be complex, difficult. It's not necessarily trivial. Now, We're going to formulate this MLE. Assume m data points, and represent to a D set, contains all data. They are independent identically distributed (iid) from unknown distribution, $P^*(x)$, which means assuming basically data are random, but all drawn from the same, true distribution, you'll necessarily know. Fitting a model to the data means that we want to find the best parameter value such that, under θ, the chance of seeing the data is going to be the maximum, called **maximum likelihood principle**. In terms of math formulation, solving the estimate for the parameter can be cast as this optimization problem. We first try to find the distribution of data under parameter θ. In fact, I would like to put this as a $\hat{\theta}$ because that indicates the estimate, that's the maximizer hopefully solve from this optimization problem. θ can choose your model, so by varying θ that you want to control, you want to find the probability, the likelihood of getting data being maxed. And we typically going to take log, but it doesn't change the optimal solution because log is a monotone function. Next, we're going to write the joint distribution Π of other data because they're all iid. We can write their joint distribution as the product of their distribution, $P(x^i|\theta)$. This is each individual samples distribution, so we will multiply this, m times. Now if you see why we're going to introduce the log, because after introducing the log, the product will become the sum. So, I can use the property of my log, write this as $\sum \log P(x^i|\theta)$. Why this is important is when I solve my optimization problem, and if I tried to maximize a product of a bunch of terms, it's going to be quite hard. When I try to find a gradient, I have used a boundary of chain rules to get my gradient vs if I maximize a sum of a bunch of variables, these sum terms are decoupled from each other. It gave rise to simple expression for the gradient. It's mostly for the ease of optimizing the function.



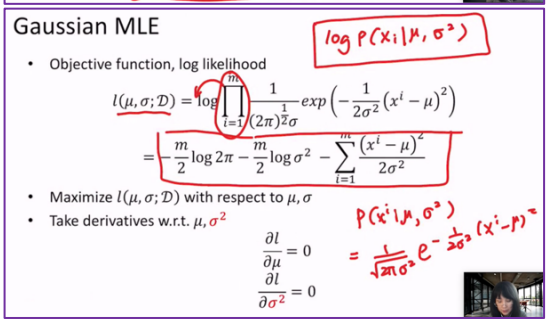
Let's see one example. What does MLE become? Think about the example of getting a θ in a biased coin. We start with a sequence of iid flips, and it represents using D. If it's a head, we're going to mark it as 1, otherwise, as 0. The θ is the probability of getting a head. First, we're trying to use our MLE prescription here. We want to find out the likelihood for each individual data points, and in this case, data is binary. It's a simple model using the Bernoulli random variable. The probability of getting a 1 is θ, the probability of getting a 0 is 1-θ. If I draw my PDF, it actually looks like this. There's a trick actually to write this expression in a more compacted form, $P(x|\theta) = \theta^x (1-\theta)^{1-x}$. You can verify these two are equivalent because when x = 0, you're going to have $\theta^0 (1-\theta)^1 = 1-\theta$. This expression is going to write a compact expression for my likelihood of single observation. The probability of observing x^i, $P(x^i|\theta) = \theta^{xi} (1-\theta)^{1-xi}$.



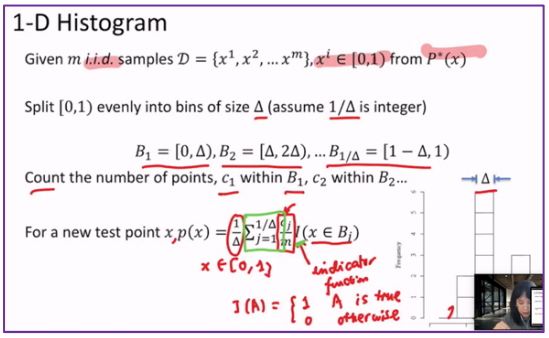
Let's try to derive the MLE. First, specify the log-likelihood function. This notation **l** denotes the log-likelihood function here. The intermediate step that we're going to specify, which is log since the data are iid. This **l**(θ;D) is going to be the sum over all the likelihood for each individual sample $P(x^i|\theta)$. Then I'm going to plug in the expression, θ^{xi} $(1-\theta)^{1-xi}$. Using the property of the log, I can bring down to x^i logθ and $(1-x^i)$ log(1-θ). I noticed that I can actually combine this because that's the term depending on i. If I combine them, what does it mean? Actually, x^i = 1 only when you have a head, so summation over x^i is the number of heads. Let me define n_h as the number of heads. The first term is n_h logθ and the second term is $(m-n_h)$ log(1-θ) because $1-x^i$ is going to be $m-n_h$. Once we have the log likelihood function, we are going to solve the maximization problem. We want to maximize this **l**(θ;D), and we going to use the same trick, taking the derivative of this **l** with respect to θ and set it to 0 and find from this a solution to θ. In fact, you can verify that. This is the expression of our log likelihood, and the derivative of this is given by this expression. From here, MLE is given by actually pretty simple expression, which is the number of heads divided by m, the total number of tosses. In the end, the maximum likelihood estimate is basically counting how many heads you have and then see what's the proportion of heads in the total number of tasks that gives you the estimate of probability of getting a head.



How do we estimate the parameters for Gaussian distribution? This can get a little bit trickier, so much illustrating the procedure. Let's consider a 1-dimensional Gaussian, this is the distribution is a bell-shaped curve and then the parameters we need to estimate is just two, which is the mean and sigma (standard deviation). Given the m iid samples, the likelihood for one data points is given by this. Here we're writing this short-handed, but keep in mind this actually also have the other constant in front of it. We can derive that the MLEs, I'm going to put a hat on it and sometimes put a MLE to emphasize these are MLEs.



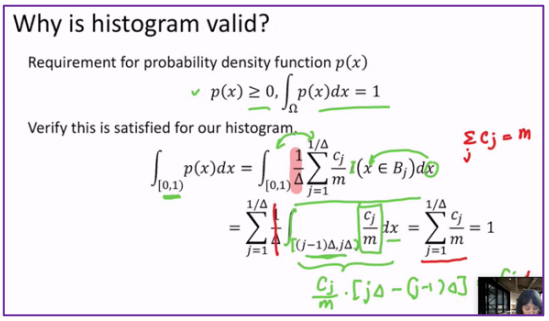
How are these expressions derived? The same thing we can write the objective function, the log-likelihood, and then since it did our iid, in the end this product will become a summation outside. So inside you how to find out log of individual samples, likelihood function given the parameters in this case. This is getting a little bit more complex when you're dealing with Gaussian. But by finding this and where $P(xi|\mu, \sigma^2)$ is given by $1/(2\pi\sigma^2)^{1/2}$ e raised to $-(1/2\sigma^2)(x^i-\mu)^2$. You plug this in there and try to simplify the expression. And after some manipulation in the end, you have this expression for your log likelihood function. The same strategy you would maximize this with respect to your parameters. To do that, I take the derivative of this and set this to be 0. From this, we can solve for the MLE, and you can show that they actually are given by this expression.



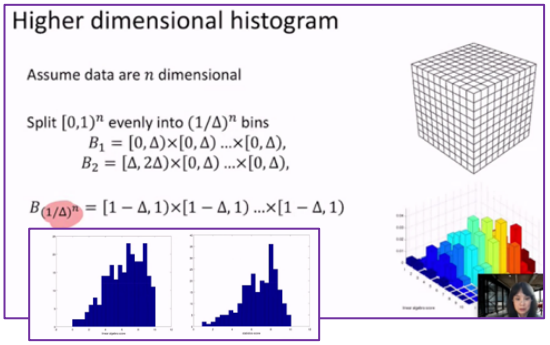
Let's start to talk about some non-parametric density estimation. We'll start with a simple 1-dimensional histogram. I'm sure everybody have seen this before as this is the default way of getting a density estimation. Suppose given m, iid samples, samples are ranging [0, 1). This is a reasonable assumption because if you have bounded observations, we can always normalize it by subtracting the center and divide by the range of your data to map it into between 0 and 1. This come from some unknown distribution P*(x), and the law, we're going to take a little bit more rigorous mathematical description of a histogram. First, we're going to split the interval [0, 1) into small bins of size Delta Δ and the number of bins, 1/Δ. To simplify notation, assume 1/Δ is an integer. Each bin is going to be represented by this interval. The first bin B_1 is [0, Δ), the B_2 is [Δ, 2Δ), and you have total number of my own work Δ bins, $B_{1/\Delta}$ = [1-Δ, 1). Histogram is nothing but just count how many points that fall within this range in each of the bins. You can see data points, c_1 within B_1 and so on. The rest is just basically representing your histogram as, how many points fall within each of the interval. If you have a continuous distribution, we have to be a little bit careful to make the PDF estimate meaningful. We're going to construct it this way as you can see, this is a fancy way of writing the continuous distribution function where this x is between 0 and 1. This is the new test point, x, could happen anywhere on this line. As you can see here, this part, c_j/m, tells the relative height of the bin in the j's bin that's proportional to how many points fall into that bin. Here this I is an indicator function. That's a pretty commonly seen notation in probability. The indicator function I(A) =1, when A is true, otherwise is 0. It's indicating some event happens. Here, the indicator function is applied on the event that your test point fall into the range of the j's bin. You're going to take the value c_j/m and scale by 1/Δ, which is particular here. I think that's intuitive and we're going to show why this is important because that's to make sure your p(x) is a legitimate density estimation. And we also have this summation term here.

But this summation term, as you can see, your point axis is only going to fall into one of the bins. This summation essentially is saying only the bin that you're falling to, the corresponding indicator function, is going to be non-zero. The other bins indicator functions are going to be 0. This summation really is only taking the value on the bin that your point falls into. So more compact way to write it. This is our PDF estimating here.

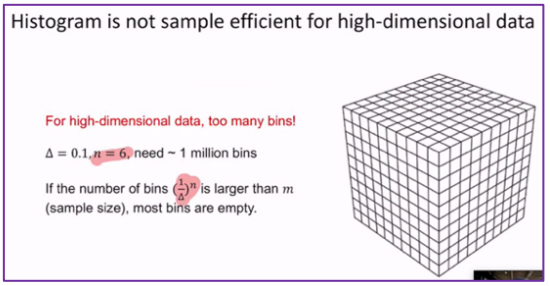
Why this histogram is a reasonable valid estimate for the PDF? Remember, the requirement for a p(x) to be a PDF is it has to be non-negative because all the counts are non-negative, and also integrate to 1. So now you will see why we're introducing this 1/Δ. That is useful to making sure this integration is going to be 1. Let me go through this equation to explain to you. This is supported on [0, 1) and we're going to plug in here the expression for p(x). What happens is, you have this integrate over x and this really controls the value of the test points. This integration happens inside, so I can change the order of summation and integration. This indicator function I is only going to take non-zero value when your x is in that j bins range, so this integration inside is going to have a smaller range by exchanging the order of summation and integration. So, you have the summation outside and this integral over the range for the j's bin inside. This c_j/m is a constant, so essentially, you're just integrating over x. This whole integration expression is c_j/m * the size of the interval JΔ - (J–1)Δ, so c_j/m times Δ. That's the value for integration. If I plug it, I have a Δ here and 1/Δ in the front, so this cancels out. After this simplification, I have this, and this actually sum up to 1. C_j is the count of the points falling into the j's bin. Because of that, the total should be equal to m, all the samples. This essentially is summing over all the bins for the total number of points falling into each of the bins. In the end, you divide this by the numerator m, and it becomes 1. This verifies that we didn't define any meaningful PDF using this histogram as the density estimation.



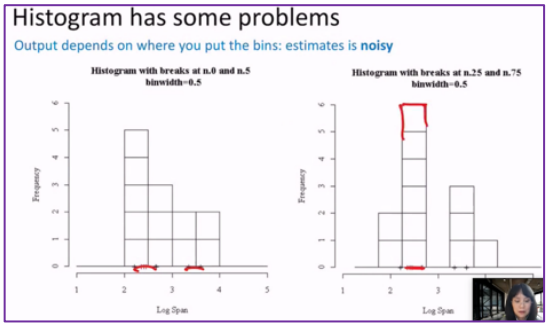
You look at these histograms which is generated using building function, and all of sudden this becomes less trivial. After these verifications you know there's math behind it. This idea can be similarly generalized to higher-dimensional data to construct higher-dimensional histograms. Suppose your data have n-dimension and assume each dimension has been normalized to [0, 1). We can split each dimension evenly into a number of bins with the same size Δ, and then you're going to end up having lots of bins by calculating each dimension. It can imagine where we are splitting these dimensions. In the end, you're going to have this cube. The total number of bins is going to be $(1/\Delta)^n$ because in each dimension 1/Δ bins for n-dimension, that's the total of bins you have.



This is a good idea to construct higher-dimensional histogram, but people don't like this idea. For visualization, if you have 2-D histogram, it is probably fine, but if you want to estimate density functions for higher dimensional data, it is just not a good idea because you have too many bins. Remember the number of bins we need for higher-dimensional data using this idea is going to be at $(1/\Delta)^n$ for n-dimension, and it will grow exponentially fast with dimension. If assuming Δ is 0.1, the size of bin and 6-D data, the number of bins is going to be on the order of one million. To fill out the bins, you need at least one million data points so that you have at least one point in each of the bin. If your data is less than a million order, most of the bins are going to be empty. That means you're going to have something not very meaningful in terms of the high-dimensional histogram. So clearly you can see the problem behind this.

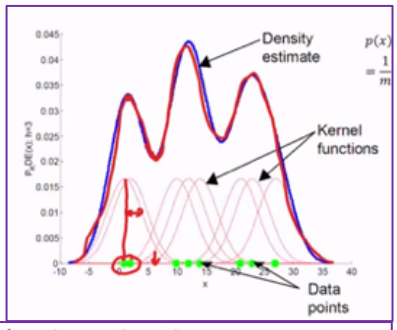


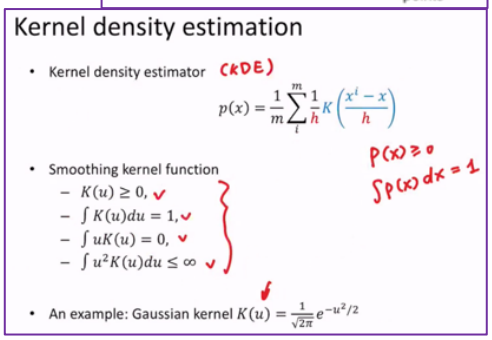
If you want to use 2-D histogram as the density estimation, it doesn't fit for high-dimensional data, and the estimates are also actually pretty noisy. For example, you look at the way you construct the bin-width, I show you to equally oscillating the bins using size Δ. But just imagining, in this case, I don't have a lot of samples and bins are not enough. So, if I shift my bins just a little bit, the points become pointing to this thing instead of being split into two bins. If you look at the shape of my histogram visually, it looks quite different, so that's a big problem. If I just shift my bin a little bit, I end up having very visually different histograms. This is not okay, especially if you don't have a lot of sample points. So how do we get around this problem?



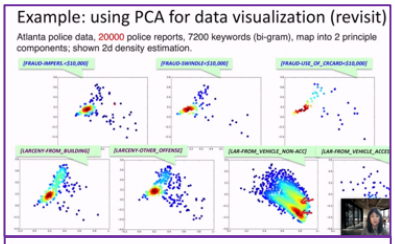
One approach is to use kernel density estimation. It's a natural idea, basically, you going to say what happens to my histogram is almost I'm going to try to approximate my density using this box shaped functions. Now, you want to make it a little bit better by placing

a smoother shape density at each of the data points instead of using a box. Here comes the KDE. That's basically formalize this idea to place a smooth kernel at each of data points. Let's look at the picture first. These green ones are the data points, and so we're going to place one smoothing kernel, a Gaussian shape kernel centered at this particular data point and decayed with a certain σ. After putting all kernels at each data points, we're going to sum up them together, super-position, and these results become this blue shaped density. You end up interpolating the data points essentially and getting this density as much as you can see. If the data are pretty close to each other and this smoothing kernel going to start to accumulate and can have a pretty high peak. If there are not many data points like in here you have a gap, and then we're going to have a smooth valley that because you don't have a kernel there and these are the tails contributions from the other data points. In the end you have a pretty smooth estimable density that's less sensitive to the choice of the bins and so on, so you have something better. This is just a formal way of writing the idea. The k is a kernel function, this argument means that I am going to place one kernel centered at each data point, and the size of kernel means how quickly in this case is going to be controlled by this parameter h, called the kernel width sometimes. The kernel needs to satisfy some constraints and properties. I need to choose a good kernel to make this is a valid probability density function, so meaning that I want the p(x) is non-negative and integrate to 1. These are the require-ments and basically the kernel function is symmetrical. If you look at the mean of the kernel is going to be 0, it has a finite second order moments thing. If one particular kernel to satisfy all of these requirements is a Gaussian kernel, it bow shape the function. We're going to sum this over all the data points and then performing the average.

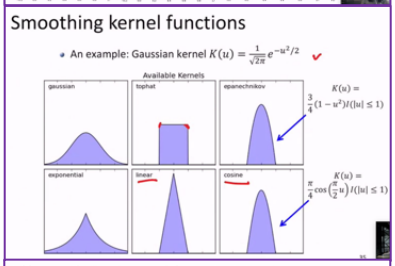




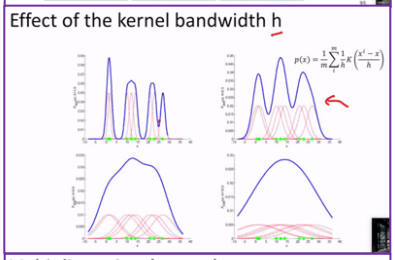
In fact, bring this back to practice. Remember we talk about PCA for data visualization. We have 27 Atlanta police reports, and then we use the bag-of-words reputation of the police reports. Each data point is the police report, and we perform the principal component analysis to project this into 2-dimension using their principal components and visualize their principal components. Instead of naively looking at a scatter plot, we also want to understand the density. My visualize of this result is generally using KDE and the color is representing the height of the density, so you can understand where the points are mostly clustered and centered.



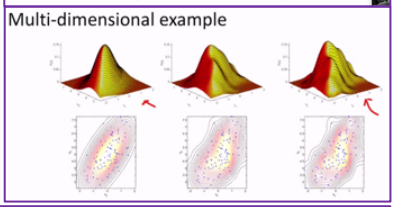
What are some possible choices for the kernels? Gaussian kernel is most common choice besides that, there's also these fancy kernels, cosine kernels, exponential, linear, and tophat (this is probably not very popular because these edges, so you won't have a smoothing kernel). These kernels have a different theoretical property as well.



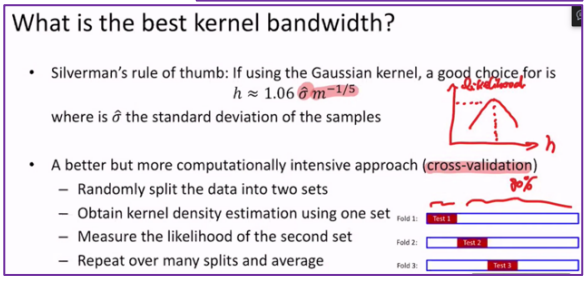
How do we choose the kernel bandwidth? It's going to determine the performance of best estimation critically. Here are some point illustrations. If you choose the kernel density to be too small, then you end up having this (first pic). You're not doing enough interpolations we're going to end up with a lot of these valleys that doesn't really tell you the shape of the distribution. If you kernel bandwidth is too large, you end up having too much interpolation, and in the end, you basically blur out everything, you couldn't see the modes, the shape of distribution. Maybe somewhere here (second pic) is a good choice of the kernel. Bandwidth h shows the shape of the distribution function well, and it is not getting too noisy, so it is enough to interpolation.



The same thing can be observed for the multidimensional case as well. As you can see it by changing the kernel bandwidth, you end up from a very smooth estimate, but may lose some detail in your density function versus something too noisy. Choose the right kernel bandwidth is quite important.

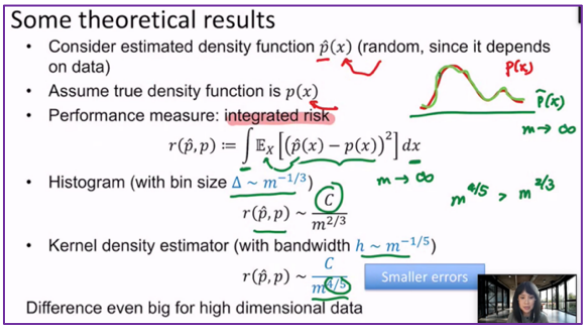


How do we choose the kernel bandwidth? This is quite important in getting good results. If you are using Gaussian kernel, Silverman's rule of thumb tells you that you can choose h to be 1.06 times the standard deviation of your samples $\hat{\sigma}$ times the sample size m raised to $^{-1/5}$. This is summarized from people's practice. A better but computationally more expensive approach is cross-validation. You can randomly split your data into two sets. For example, I have five different ways to split my data and use 20% data for testing and 80% data for training. For each fold, I'm going to estimate the kernel density using training data and measure the likelihood of test data. I'm going to repeat this many times alternating which segment of data is going to use for test, and this is going to

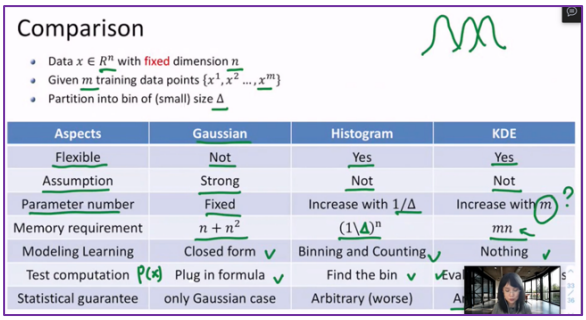


give a validation of how good my kernel bandwidth is entered. I can get the function like this by varying my kernel bandwidth twice, then corresponding average likelihood. You want to see this kind of pattern. Somewhere in the middle, you have optimal choice of the bandwidth that gives you the largest likelihood on the test data.

Here is a simple demo for the KDE and histograms on the wine data set that is the result of a chemical analysis of wines grown in the same region of Italy but derive from three different cultivars. There are 14 features, and I guess different components probably determines how good wine taste. Let's try if we can see from the data. First, read the data. Since we have a multi-dimensional data, we're going to perform PCA to project the data into 2-D. Here choose d as 2, find the covariance, extract top two dominating the eigs matrix, project my data, and get the principal components. Now I'm going to try to get a histogram following the steps instead of calling function: map the data into a number of bins, here I use 10 bins, and then start to count how many points fall into the bin. I'm going to plot the height of that. This is the histogram of the values of the features for the first principal component. Notice that you have to divide this by the size of the bin to make it a real density estimate. And then this is for both the first and second components and going to be 2-dimensional results. Here it shows some distribution of the data along these 2-dimension. Next is performing the KDE. We're trying to place one kernel function at each data point using a Gaussian kernel with the bandwidth to be one. Here is the KDE results. Let's see if this compare with my 2-D histogram. You can see it's quite visual and this actually have three components versus looking at here. From 2-D result, it's harder to see the components, but KDE clearly illustrates three components in my density.



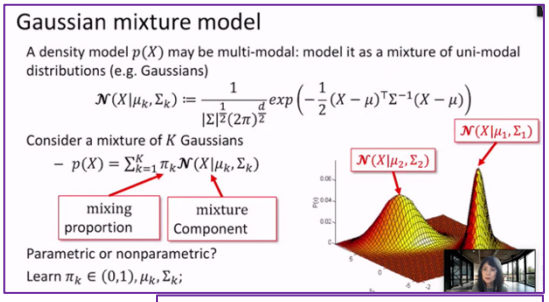
Now, I would like to present a little bit of theoretical results in this problem to understand the performance of our density estimation. We're going to assume the estimated density function, $\hat{p}(x)$. This is actually random because it depends on the training data, which is modeled as random, so estimated themselves. The idea is basically we have the true density function, p(x) and an estimated density function $\hat{p}(x)$. And we want that the estimated one is pretty close to the true one, especially when the number of samples becomes large, hopefully they're getting smaller and eventually come to 0. First, we introduce a measure of the performance called Integrated Risk. I'm going to look at the pointwise difference between these two true and estimated density functions, then take the square of the difference to accumulate this over all the possible values of x. There's one thing that's a little bit tricky is that I also have to take the expected value Ex of the square of the difference of x. Why do I have expectation here? That is because $\hat{p}(x)$ is random. This Ex is with respect to the randomness in $\hat{x}$ considering the distribution of my data and then how the $\hat{p}$ will distribute according to this data distribution. So, an integrated risk is my performance measure. It has been shown that when the bin size for each histogram is chosen to be on this order, $m^{-1/3}$. That means if you have more samples, you can shrink the size of your bin for histogram, and then you can achieve the order of integrated risk to a C divided by $m^{2/3}$. You will say there is a constant C. What does it mean? Especially in this analysis, we're going to pass the m to be large, typically goes to infinity. That means is constant C doesn't depend on m. This basically says the performance integrated risk will decay as this outer $m^{2/3}$ versus the KDE. It has been shown that if h is $m^{-1/5}$, the Integrated risk is on the order of a C divided by a different polynomial order. Since $m^{4/5}$ is typically larger than $m^{2/3}$, the decay rate of the KDE as sample size becomes large, is going to be faster. In that sense, current density estimation is better because you can have smaller risk for the same number of samples.



Besides that, we are going to talk about these different methods. What are the comparisons in terms of different considerations? I offer this table here. Suppose you have a data with n-dimension, and m training data points. If I'm talking about histogram, then the size of the bin, Δ, we typically have to choose Δ to be small. In the previous slide, we have to choose the Δ to be smaller as you're getting more and more samples. In terms of the flexibility of the model, Gaussian is most strongly assumption. You have to impose a parametric form and the distribution is a bell-shaped curve decay, so it's not flexible. That means if your data doesn't follow a Gaussian distribution, you're going to suffer performance loss because of model mismatch. Versus the histogram and KDE, these non-parametric methods are more flexible, and these are not strongly assumption. There is a tradeoff because the number of parameters is fixed in Gaussian, the parametric model. It means it doesn't increase with number of samples (KDE) and it doesn't increase as you shrink the size of your bins (Histogram). In terms of the size of the model, Gaussian is more compact because of using fewer parameters to represent your data, so if your data can be well approximated using a Gaussian, Gaussian could be a good choice. KDE seems to be pretty good, but there's one drawback I want to point out is that you have to place one kernel at each of the sample point, that means to represent your density function, you have to keep all the data in memory, which is not good thing. If you have a lot of data like millions of data, do I really want to store all the data just to represent the shooter function? It's probably better to summarize the data instead of keeping all the data in the memory. There is a way to hopefully get around us in some cases. In terms of memory requirement, Gaussian depends on the parameter $(n+n^2)$ of the problem. If the number of samples is large, then KDE may not be memory efficient (mn), and the histogram have to depend on the size of the bin $(1/\Delta)^n$ and it can be pretty efficient in a sense because you can aggregate your data without having to remember everything about the data. Histograms have that

benefit in terms of the model learning. So, how do we get an estimate? In Gaussian, we have the closed form, the mean, and the covariance matrix, or the MLE for the parameters. For histogram is pretty simple, you just find the bin and count how many points fall in there. I said nothing for KDE because, there really isn't anything you have to fit to the model, the data plays the kernel, and that's it. The test computation is if I have a x, how do I decide/compute the value of p(x)? Gaussian, you plug in the formula because everything is pre-specified. In histogram, you have to find the bin and corresponding height of the bin. The most expensive one is KDE to evaluate m functions. Those functions are the kernel for example, Gaussian kernel that we have to evaluate the value of x at each one of the kernels, so there are m kernels corresponding to the same number of samples. It's pretty expensive in that sense. There are some statistical guarantees we haven't explained, and KDE has better performance guarantees. Basically, KDE is flexible, but a fairly expensive method and there are some limitations. If the number of samples is large, this may not be a good choice.

Here is a little bit peek into the answer I'm trying to propose in the next slide, in the next lecture. If you have a lot of data and you're trying to, for example, using KDE, you're placing one Gaussian kernel at each of the data. But for example, you can clearly see that the data seem to have 3 modes in the wine data example. That means you probably don't need to use many Gaussian kernels at each sample location. If you know there are a few modes, maybe it's better to directly fit a number of Gaussian components to your data. That's the idea of a Gaussian mixture model. We'll give more details about this, but a high-level idea is we're trying to approximate the distribution of the function using a mixture of Gaussian. In fact, the Gaussian mixture model is quite flexible, and it's somewhere between parametric versus nonparametric model, it's parametric because you still have a number of parameters, but it's also nonparametric in a sense that we can use this to approximate pretty general shapes of distributions. This is an industry, if you do observe clusters, structuring of data maybe can directly fit Gaussians.



In summary, we have talked about density estimation in this lecture, we have talked about both parametric and non-parametric ways of estimating densities. And we discussed the pros and cons and tradeoffs of these methods and did a comparison and also leave away for the next lecture. How do we estimate a general Gaussian mixture model? Okay, that's it for today.