COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

Week 3
Video Transcripts

Video 1 (08.36): Bayes Linear Regression 1

In this lecture I want to continue this line of discussion of how Bayes rule relates to the least– the linear regression problem. So let's look at the Bayesian linear regression problem. So, again, to refresh our memories we have the vector 'y' and 'Rn'–this is the vector of responses, and a matrix of covariance–of covariates, that's 'n by d', where the ith row of 'y' and 'X' correspond to the ith observation pair 'yi', 'xi'. And we're not really worrying about whether, there's an additional dimension of ones or whether we've standardized the data. We assume that we pre-process things in a way such that it works out. We're more interested in the math techniques here. So in the Bayesian setting, we use the following model for this data.

We assume a likelihood, we assume that 'y' is generated from a Gaussian with mean equal to 'Xw' and covariance Sigma squared 'I'. And we then, place a prior distribution on the vector 'w', which is zero mean Gaussian and the covariance Lambda inverse 'I'. So in this case, the unknown model variable is the vector 'w' in 'Rd'. And, we are hoping to learn something about this vector. And so what the likelihood model does here is it says how well the observed data agrees with a particular vector 'w', and the model prior is our prior belief or our prior constraints that we're going to place on 'w' that we're going to define in advance. So we call this Bayesian linear regression because we've defined a prior on an unknown model parameter or a model variable, and we're now going to try to learn the posterior distribution of the model variable, given some data. So again, we saw last time that instead of finding the full posterior, we could find the MAP solution, where we returned the maximum of the posterior distribution of 'w' or equivalently the maximum of the joint likelihood of 'w' because we saw that the normalizing constant here, did not actually impact the location of the solution. Start that over. Okay. So before, discussing Bayes rule in the context of linear regression, we saw last time how the Maximum A Posterior solution can be found, also called the MAP solution, by finding the value of the vector 'w' that maximizes a posterior. And we also saw that because the– by Bayes rule we can eliminate the normalizing constant, we can equivalently say that we want to return the value of 'w' that maximizes the joint likelihood of the data and the model variable, both of which we know. So again, I was able to get rid of this term, previously, because it didn't actually impact the location of the maximum.

Okay, so last time saw that we could find the MAP solution using Bayes rule and using the posterior, where we find the point of the vector 'w' that maximizes the posterior, so equivalently the point of the vector 'w' that maximizes the log of the posterior. And that solution was the same as the vector 'w' that maximizes the log of the likelihood, plus the log of the prior, in other words, the log of the joint likelihood because the product of the likelihood and the prior is the joint likelihood. When we solved this for the Bayes for the linear regression problem, we saw that the MAP solution was equal to this, which corresponded to the Ridge Regression solution, where we've made a redefinition of that regularization parameter. Okay. So this, brings us to the difference of point estimates for model variables and Bayesian inference. So point estimates, what we mean when we say we find a point estimate is that we find a specific value for an unknown model parameter, according to some inference or object–inference procedure. So for example, the MAP solution and the maximum

likelihood solution, corresponding to Ridge Regression and least squares are both a point estimates, because they return specific values of the unknown model variable 'w'. So, Bayesian inference instead of returning a point estimate, is going to try to define or return a probability distribution on the unknown model variable, in this case, 'w', given the data. So Bayesian inference, instead of returning a specific 'w' is going to return a distribution on 'w'. So let's look at Bayes rule and the linear regression problem that we've been discussing this far. So since, 'w' is a continuous valued random variable in 'Rd', Bayes rule says that the posterior distribution of 'w', given data 'y' and 'X'–so that's what this is, the posterior probability of 'w' given data, is equal to the likelihood of the data, times the prior on 'w', divided by the normalized– the normalizing constant, which is the integral of the numerator over all possible values that 'w' can take. So that is, we get an updated distribution on 'w' through the transition of prior to likelihood to posterior. Or, in other words, we would say the posterior probability of something is proportional to the likelihood, times the prior.

So let's see, how we can do fully Bayesian inference for the linear regression problem. So again, we're going to work with the proportionality. So the posterior distribution of our regression coefficient vector 'w', given the data, is proportional to the likelihood of the responses, given 'w', and given the covariates 'X', times the prior of the regression coefficient vector 'w'. We've defined the likelihood to be a Gaussian, a Multivariate Gaussian with this form, and the prior to be a zero mean Gaussian with this form, where again because we're only working with proportions, we can eliminate anything that premultiplies this function that doesn't involve 'w'. So we removed some terms here in the proportionality. And then if we, multiply this thing through, we can equivalently say that the posterior is proportional to this term. So what I have done here?

I've actually expanded this into the three terms that result from taking this product. I then, brought this term over into this exponential to have one exponential term. And then, I've reorganized some things so that we we can write is a quadratic 'w', where this term results from the prior and this term results from the likelihood, as well as this additional term here. And then the term that involves 'y' transpose 'y', again I've removed that because that's simply equal to a premultiplication of this value. And so, when we normalize this, when we divide by the numerator integrated over 'w', this will simply cancel out. So again, we take advantage of the proportionality that we're working with to remove terms that aren't necessary.

Video 2 (05.29): Bayes Linear Regression 2

Okay, so where we're at now is that we need to normalize this function. We need to divide this function, by its integral overall values of 'w' in 'R^d'. So there's two key terms here. There is this quadratic term in 'w', and there is this linear term in 'w'. And so already from this we conclude– we are able to conclude that, the posterior distribution is Multivariate Gaussian. So in the next slide, I'll go through some steps for actually doing this, but first, why is it that we can make this conclusion? So, in order to solve Bayes rule, we're allowed to multiply and divide this term by anything that doesn't involve 'w'. So we can multiply it by something, or divide it by something, not involving 'w' such that we know that the result integrates to one and we've solved the problem. So we know that a Gaussian has this form in the exponent. We know that the Gaussian has minus one half times, this term, in its exponent. And so what we can do is complete this square by adding terms that don't

involve 'w', in order to, get this quadratic to be written in this form. So in other words, let's compare a Gaussian distribution with what we have.

So here's what we have so far. We want to divide this term by some number 'z', in order to have a probability distribution. So we want to find the value 'z' such that, this function integrates to one. And, on the other hand, we know that a Multivariate Gaussian with mean mu and covariance Sigma has this form. So what I've done here is actually multiply out, this quadratic term from this slide. So we know that a Gaussian looks like this, with mean mu and covariance Sigma. Here's where we're at right now. We want to find this value of 'z', that makes this function integrate to one. And so immediately, we're able to conclude that if we define this–for this Gaussian, if we define Sigma inverse to be equal, to this, and we define Sigma inverse times mu to be equal, to this, so we plug in this term for Sigma inverse mu here. Plug in this term for Sigma inverse up here, which means we have to do something clever here.

We have to say mu transpose Sigma inverse times Sigma times Sigma inverse mu. So we plug the inverse of this in here, and we plug this vector in, both here and here. Then we'll immediately find that if we set 'z' to be equal to this, that these two, are equal to each other. And so this is just a quick way, that we can calculate the posterior as being Gaussian without actually ever having to try to solve this integral over 'w'. We can say for 'z', plug in this, and then solve and find that we can simplify the result to be a Gaussian with covariance, equal to, the inverse of this and mean equal to an–the inverse covariance times the mean equal to this. In other words, what we can say is, that the posterior of 'w', given data 'y' and 'x' is a Gaussian with mean equal to this, and covariance equal to this. And again, if we write out this function, where we plug this in the covariance, and we plug this in for the mean, we expand everything out, we'll find that we can go back here, and it's equal to, this function divided by, this constant with respect to 'w'. So in other words, a Gaussian with mean and covariance set to be equal to, these two things is proportional to this function.

We simply need to divide it by some constant with respect to 'w'. Let's look at a couple of interesting things about this. First, notice that the mean solution, the solution for the mean of the posterior distribution, is equal to the MAP solution. So, if we use Bayes rule to calculate a posterior distribution of 'w', the mean is going to be equal to the MAP solution, which we also know is equal to the Ridge Regression solution. But now we have a covariance Sigma which captures our uncertainty about 'w', in a way, that's the variance of the Ridge Regression and least–squares solution also captured some uncertainty of 'w' but now, we actually have a functional distribution that we can work with. So we can actually give densities and calculate probabilities that we couldn't do, when we calculated these variances.

Video 3 (10.20): Uses of Posterior Distribution

Okay. So, let's look at some uses of this posterior distribution. So, on this slide, I give two very simple examples of something we can do with the posterior 'w'. One is we can give probabilities. We can ask questions like, 'Is the i^{th} dimension of 'w' positive or negative?' And the reason why this would be important, is that the i^{th} dimension of 'w' says that by increasing the i^{th} dimension of 'X', we increase the expected output, whereas if 'w_i' is negative, by increasing the i^{th} dimension of 'X', we decrease the expected output. So this tells us how 'X' relates to 'y', and so this could be very important for practical purposes. Or, can we confidently say that 'w_i' is nonzero? And so, all of these can be

calculated using the marginal distribution on the i^{th} dimension of 'w', which under the posterior is just a Univariate Gaussian with mean equal to mu 'i'–so the i^{th} dimension of the mean and the variance equal to the i^{th} diagonal element of the covariance matrix, Sigma. So this is something that we can show. And we can use this to characterize what we think the true value of 'w_i' is, whether it's positive, whether we're confident that it's positive, whether we're confident that it's negative, or whether we can't even say whether it's nonzero.

Also, we can ask questions like, 'How did the i^{th} and j^{th} dimension relate?' So again, we can calculate the marginal distribution of the entire vector 'w', by looking at the probability distribution restricted to the i^{th} and j^{th} dimensions. That's also can be shown to be a Multivariate Gaussian, where the mean is simply the corresponding dimensions of the mean-vector, and the covariance is the submatrix form, by taking the corresponding dimensions from the matrix. So these are things that we can calculate, in order to understand what we think the values of the regression coefficient vector 'w's' will take and how they relate to the covariates 'X'. But also important is–and perhaps more important, which I want to discuss next is, the procedure of predicting new data, given the old data. So we can use posterior to help us make predictions for new data. So let's look at how we can make predictions for new data, using Bayesian linear regression. So let's recall that we have a new pair 'x_0', 'y_0', where 'x_0' is the measured covariates. So we get 'x_0' and we want to use 'x_0' to now, somehow predict the unknown response variable 'y_0'. So we saw how using the least squares or the Ridge Regression solutions, which correspond to the maximum likelihood or the MAP solutions– what we simply do is say that we believe 'y_0' is approximately equal to the dot product between 'x_0' and either the least squares solution or the Ridge Regression solution.

In both cases, we have a point estimate of 'w', and so we have– and so we give a point estimate of our prediction, with no uncertainty about what we think 'y_0' is. So with Bayes rule, we can make probabilistic statements about 'y_0'. What we want to calculate is a probability distribution on 'y_0', given 'x_0', and given the previously observed data. So this is the probability that we're interested in calculating. And so, how can we actually calculate this probability. Well, we can represent that as a marginal probability over the unknown coefficient vector 'w'. So we include 'w' in a joint distribution, and then, immediately integrate it out to get the marginal distribution. And then, given this joint distribution we can represent it as a conditional distribution on 'y_0', given w, and given all of the observed data, times this conditional distribution on 'w', given all of the observed data. So we can factorize this joint distribution into the product of these two conditional distributions.

This is just simple manipulations of probabilities. However, notice first, that conditional independence here, let's just say that the probability of 'y_0', given the vector 'w', given the covariates 'x_0', and given all of the previous observations, is just the probability of 'y_0', given 'w' and 'x_0'. Why? Because of our conditional independence assumptions. So our assumption of how 'y' is generated is that, if you tell me the corresponding 'x_0', and if you tell me the regression coefficient vector 'w', I have all of the information necessary to define the probability distribution on 'y_0'. I don't need to take into consideration any other data. It's simply a Gaussian with mean equal to the dot product between 'x_0' and 'w', and variance equal to Sigma squared. So that's the likelihood assumption that we make. And also, the probability of 'w', given 'y', and given 'X', and given also 'x_0' can be shown to be conditionally independent of 'x_0'. So if you tell me 'x_0' but, you don't tell me 'y_0', then I have no extra information in order to update my probability of 'w' above and beyond 'y' and 'X'. And so, this distribution is equal to the probability of 'w', given 'y', and given 'X', which we observed is the

posterior probability distribution. So both of these distributions are things that we know. The likelihood distribution–so this distribution is inserted here is the likelihood, by the definition of the model, we can write that, and this distribution is replaced by the posterior distribution because of the conditional independence. And, we've already discussed how we can calculate that through a linear regression problem. So this has a name. It's called the predictive distribution. So we're predicting new data, given old data, and we represent the predicted distribution as a marginal over the likelihood of the new data, given model variables, times the posterior distribution of those model variables, given the old data. And so intuitively what we're doing, is we're evaluating the likelihood of something new for a particular model variable 'w', and then we're weighting that particular model variable by our current belief of what we think is going on with 'w' as represented in our posterior distribution.

We then, sum all of those possible weighted models together. In this case, because there is an infinite number–continuous and infinite–continuous number of possible vectors 'w', we're integrating. So we integrate over all possible model vectors 'w', and we're essentially removing all of the uncertainty of 'w' in our forming our predictive distribution. Okay. So let's look at predicting new data for the linear regression model. So again we have the model of the new data, given the model variable–is a Gaussian like this with mean and covariance defined to be this. And by Bayes rule, we know that the posterior distribution of the regression coefficient vector, given 'y', and given 'X' is a Multivariate Gaussian with mean mu and covariate Sigma, where I have shown how to calculate those on the previous slide. So I won't work through the integral but, what we can show by calculating this integral using the Gaussian representations for these distributions, is that the predictive distribution of 'y_0', given 'x_0', given 'y', and given 'X' is also a Gaussian with mean mu_0, and variance $Sigma_0$ squared, where the mean is equal to 'x_0', the dot product between 'x_0' and the posterior mean of 'w'. And the variance is equal to the noise variance of the model. This is the noise from the likelihood, which we can't ever eliminate, plus some additional variance, which is equal to the product of 'x_0' with the matrix Sigma. So notice, that with this predicted distribution what we're doing is we're saying that our mean of the Gaussian– of our predictive Gaussian is equal to the MAP prediction, because remember that we showed how the mean of the posterior is equal to the MAP solution. And so, our mean–the mean of our predictive distribution is the MAP prediction, the point estimate we would make under MAP. But now, we have this additional variance, which is used to give us a sense of how confident we are in what 'w' is, which we couldn't have before because we used a point estimate to solve for it. Whereas now, we have the posterior distribution of 'w', and so our uncertainty of 'w' is contained in this distribution, and we can then use that uncertainty to propagate forward to the predictive distribution.

Video 4 (08.40): Active Learning 1

Okay. So, we've discussed Bayes rule for the linear regression problem and predicting new data. Let's now, look at something that combines both of these two techniques, something called, active learning. So active learning is a general technique for learning model efficiently, and we're going to discuss it in the context of Bayesian linear regression. So, Bayesian modelling can be thought of as a sequential process, where we see a sequence of data, and given everything that we know up until a certain time point, we calculate our posterior belief of the model variables. We use that to then, make a prediction for a next observation. And after then, seeing the next observation, we update

our posterior belief. And so the idea, of active learning is to do this efficiently. So imagine that we have options, we have choices, in how we measure the data, or which data we observe next. Is there a sequence that we can follow to somehow observe data that is going to give us a lot of information about– towards the end goal of calculating our posterior on the model variables? So for Bayesian linear regression, what we have is something like this. We have that if we let 'y' and 'X' be the old data, and 'y_0' and 'x_0' be some new data, by Bayes rule, we can calculate the full posterior, given all of the data, as being proportional to the likelihood of the new data 'y_0', given the model variable 'w', and given the 'x_0' that we observed, times the posterior of the model, given all of the old data. And so, we see that Bayes rule can be used in multiple ways.

We don't necessarily have to make this likelihood over all of the data, and this prior be the original prior. We could also make the likelihood over the last observation and the prior in that context becomes the posterior, given all the previous observations. So for the linear regression model, we have a posterior that looks like this, where we can see how we make a very simple modification to the posterior, given the new data that we observe.

So the posterior covariance, can be written in this way, where I've intentionally separated out the next observation. And the posterior mean can be written this way, where, again, I'm pulling out and separately writing the new observations that we have, whereas this is, for example, a sufficient statistic that we just keep the cumulative sum of outer products, and the cumulative sum of these vector scalar products. And we treat this as a single vector that we then update in this way, and this as a single matrix that we update in this way. Okay. So let's, now look at how we can use this sequential Bayesian learning to learn the model posterior more efficiently. And I should say that this is only really useful in the context, where we really do have a sequence of observations that we get to choose from. So, if the sequential aspect of learning does apply to the problem, then this will be useful. But if we have all of the data, already, then this isn't necessarily something that we'll want to do. Okay. So, we've seen how learning 'w', so learning a posterior of 'w', and then making predictions for the next response is really a two-step procedure. So, what we do is we first form the predictive distribution of the response of the new– that corresponds to the new covariate vector that we measure 'x_0', given all of the old data. And then, using that measured response, we update the posterior, as I've shown in the previous slide, by a rank one update, or adding a vector to a sufficient statistic. So now, let's think of how we can efficiently learn the posterior, when we get to choose the sequence of measurements or of observations that we have. So for example, imagine that we have a data set of covariates, 'x_1' through 'x_n'.

So we've gone into a field and we've measured many different locations in the ground, and so, we have some sort of–perhaps a frequency response for n different locations. And now, we want to pick which sites are we going to dig so that we can measure the actual response, whatever it may be, in order to learn our posterior more efficiently. And you can see why, this would be a useful thing to do.

For example, if measuring a 'y_i' that corresponds with–to an 'x_i', somehow costs money, maybe you have to destroy something, maybe you have to dig a big hole in the ground. So, it's somehow expensive to observe this 'y_i'. We want a way of sequentially picking these responses, so that we're going to get a lot of information about our posterior of 'w'. So first, let's intuitively think of an active learning strategy that we could do to perform this task. So this is, just a heuristic, and then I'll make it more formal in the next slide. So imagine that we already have a measured dataset, 'y', 'X' and

therefore, we've computed the posterior distribution of 'w', given 'y', 'X'. We can then construct the predictive distribution for all remaining 'x_0' in our dataset. So these are all the 'x_0', in the dataset, for which we don't have the corresponding response. And so, we've seen how that predictive distribution can be written this way.

The distribution of the response associated with an 'x_0', given a 'y', and given an 'X', can be written as a Gaussian, where the mean is the dot product between 'x_0' and the posterior mu. That's the posterior distribution here, and the variance of our prediction is written as the noise variance, plus this quadratic product of between 'x_0' and our posterior covariance, where the post–that covariance is from the Gaussian posterior of 'w'. So, for every 'x_0', we can calculate these two values, and the value of $Sigma_0$ squared, in a way, it gives us a measurement of how confident we are in our prediction. So for different 'x_0', we're going to have different measurements of how confident we are in our prediction of mu_0, in a sense. So, this suggests the following procedure. First, form the predictive distribution on all data in our set that do not have the corresponding response. Then, pick the data point 'x_0' for which our uncertainty is the greatest, and measure the corresponding response. So, go then–actually decide to go into the field and dig a hole in the ground or perform a lab experiment that might cost some money. Then, update the posterior distribution of 'w' using the new augmented dataset, where 'y' is the old data augmented by the new response, and 'X' is the old data augmented by the new covariate vector. And then, return to one and continue this process, where at every step, we're choosing to measure the response that corresponds to a covariate vector that we're the least certain about in our prediction. So this intuitively, makes sense.

It seems reasonable but, now the question is what is it that we're trying to optimize when we do this.

Video 5 (08.42): Active Learning 2

In order to, say what exactly we're doing when we perform the sequential process, we need to define some sort of an objective function to be able to say, what is it that we're trying to maximize or minimize by performing the four steps from the previous slide. So to this end, we're going to introduce a concept, a very general concept called, entropy. And, we'll only discuss it in the context of active learning, for this specific Bayesian linear regression problem. So let 'p(z)' be a probability–continuous probability distribution. Then, by definition, its differential entropy can be written in this way. So this is a function, of some distribution on a continuous random variable 'z'. This is the differential entropy of that distribution. And so, what this quantity measures intuitively is the amount of uncertainty in the distribution, so the spread of the distribution. So a larger value of the differential entropy, in a sense, corresponds to a distribution that has a smaller variance. It's a distribution, where we're more confident in the region of values of that as you can take. Whereas, when the differential entropy becomes more and more negative, then we're less uncertain about 'z', and in a sense, we can think of this distribution as having a very large variance.

So, for the Multivariate Gaussian, the differential entropy can be written this way. It's a function of the dimensionality of the vector 'w', times two Pi, times 'e', times the determinant of the covariance of the Gaussian distribution, and those terms are then passed to a logarithm, a natural logarithm function. Okay, so to analyze the procedure on the previous slide, let's first introduce the concept of

the entropy of a distribution. So, imagine that we have a continuous random variable 'z' and a probability distribution on that variable.

Then, the entropy of that distribution, or we can call it the differential entropy, because it's a continuous distribution, is written in this way. So, intuitively, we can think of this distribution as a measure of how much the distribution is spread out, so how uncertain are we about the values that 'z' can take. So, larger values of this differential entropy correspond to more uncertain distributions. So distributions that have larger variance will have a larger differential entropy. And as the variance gets smaller and smaller, the differential entropy will become more and more negative, to the point where as the variance goes to a point estimate. So you have no uncertainty, the differential entropy goes to negative infinity. So we're going to use this measure of differential entropy for a Multivariate Gaussian, in which case we can calculate that the differential entropy of a Multivariate Gaussian with mean mu and covariance Sigma is equal to this function. So it doesn't depend on the mean. It depends on the dimensionality of the original–of the distribution, so the dimensionality of 'w', and also the determinant of the posterior covariance of our distribution on 'w'. Okay, so we can use the differential entropy to show that the active learning procedure I've described previously is a greedy algorithm, whereby we are picking the covariates to measure their associated responses that give us the most information about our posterior belief of 'w'. And in this case, information is measured by the differential entropy of our posterior distribution. And so, the less uncertain we are of 'w', the more and more negative our differential entropy of our posterior distribution of 'w' will be.

So what we're going to, quickly, show is that the previous algorithm is a way of picking a point to measure that's going to minimize the posterior differential entropy among all of the options that we have. So, a way to see this is to look at how the posterior covariance, which is all the differential entropy, depends on changes when we add a new measurement. So we've seen that the prior distribution for the next observation, in this case, the prior distribution is the posterior distribution after observing the previous observation, so our current posterior distribution gives us a prior for what we think 'w' is for our next observation, is equal to Sigma, which has this functional form. And then, after we measure the response for 'x_0', our posterior covariance is updated in this way, where we have our posterior inverse covariance, plus this term, which only depends on our next measurement, that quantity inverted. And so, an interesting thing we can see is that our posterior covariance does not actually depend on what the measurement is, so there's no 'y_0' or no vector 'y' in any of these calculations. So we don't actually need to know what the measurement is, in order to calculate the posterior covariance. And therefore, we don't need to actually know what the measurements are, in order to calculate the posterior uncertainty of the vector 'w'. So, using a rank–one property of the determinant, it can be shown–and I'm not going to derive it on this slide but, it can be shown that the posterior differential entropy relates to the prior differential entropy in this way.

The posterior differential entropy is equal to the prior differential entropy, minus this term. And so, notice that we've written the update to our differential entropy of the posterior as a function of the prior covariance of 'w'. So, this is the covariance calculated using all of the measurements we have up to this point, and the location or the covariate vector that we use–that we choose to measure its associated response. So from this, what we can see is that if we want to pick the vector 'x_0' that minimizes this posterior differential entropy, we really need to pick the vector 'x_0' that maximizes this quadratic product, because this is a constant, it's not going to change. And this term, we want to

maximize this term or minimize the negative of that term. And, we can do that by picking 'x_0' that maximizes this dot product–this quadratic product. And so we see, that exactly the rule that we had discussed previously, that we pick the vector 'x_0' for which our predictive uncertainty is the greatest, which was this value.

This was our predictive uncertainty for 'x_0'. Viewing it from the perspective of minimizing our posterior uncertainty, so–trying to pick the maximum of our predictive uncertainty, viewed from the perspective of minimizing our posterior uncertainty, results in identically the same rule. So in this sense, we can see that we have a greedy algorithm, because we are only picking the next point to minimize, in a greedy way our objective function, where our objective is the posterior uncertainty of our distribution of 'w'.

Video 6 (04.49): Active Learning 3

Okay, so we discussed active learning as something, that Bayes rule, and the Bayesian approach allows us to do, fairly easily. Let's also look at something called model selection, which is something that Bayesian inference, can also handle fairly easily. So, in the context of Bayesian linear regression, when we say model selection, what we're going to mean is selecting the vector, Lambda. So, remember that our prior distribution on, 'w', is a zero mean Gaussian, where the precision matrix, so the inverse of the covariance is, Lambda times the identity. And, we needed, to select the parameter, Lambda. So, this is a nuisance parameter, in a sense, that we have to set it, or we're going to set it—we choose to set it.

But, what we set it to, can impact the performance significantly, potentially significantly. And so, let's see how Bayes rule can give a principled way, to pick Lambda using something called, evidence maximization. So, this is one thing that we can do to select Lambda, among others, for example, we've already discussed, cross validation. So, this is how we can select Lambda, using a Bayesian approach. So, we've seen from Bayes rule, that the posterior of the vector 'w' given the data, and now, I'm going to also condition on the hyperparameter Lambda, is equal to the likelihood of the data times the prior. And so, there we can see that the prior is dependent on Lambda, divided by the evidence, which is the distribution of the data, given the covariates, and given the hyperparameter Lambda with 'w' integrated out. So, in a sense, this evidence, gives us a measure, of how good our model is, and how good our hyperparameter assumptions are. So, let's see how we can use the evidence to select Lambda. So, one way that we can select Lambda is, to maximize the evidence. So, we can pick Lambda to be the arg max of the log of the marginal likelihood of the data, where we've integrated out all of the unknown model variables, in this case, that's just the vector 'w'. So, in this case, we can show— I'm not going to derive it, but we can show, that the distribution of 'y', is a Multivariate Gaussian, with zero mean, and covariance equal to Sigma squared 'i', which is the noise, plus Lambda inverse times 'X' transpose 'X'. So, that's something that can be derived. And so, now what we want to do is to pick the value of Lambda, that maximizes this distribution, or maximizes the log of it equivalently.

So, we notice that this actually kind of looks like maximum likelihood, which we've discussed previously. And in fact, it is. So, what we discussed previously could be called type one max—maximum likelihood, where we're maximizing the likelihood of the model over the main parameters. So, in this case, the vector 'w'. By maximizing this marginal likelihood, or by doing evidence

maximization, where we integrate out the main parameter, we're doing what's called type two maximum likelihood, which is called, Empirical Bayes. And so, the difference between these two is purely in their perspective. Really, it's all maximizing a likelihood over an unknown parameter, but in the first case, we maximized it over, what we called, the main parameter, so there was no distribution on that parameter, and we didn't think of that parameter as being a marginal, where we've integrated out anything else. So, that was where we maximized 'y', given, 'x', and 'w' over 'w'. Now, we're maximizing the evidence, which is a marginal likelihood, where we've integrated out the model parameters. So, for this particular problem, I won't discuss the algorithm for it. We can't—we can't do this in closed form, we would have to do some sort of a gradient method. But, I—I'm, more interested here, in introducing the general technique.

Video 7 (05.24): Active Learning 3

Okay, in this lecture, I want to continue, with linear regression, but let's now leave the Bayesian world, and let's leave squares, and Ridge Regression, and let's think about some more things that we can do with–for the linear regression problem. So, primarily in this lecture, I want to discuss what can be called, under determined linear equations. So, in this case, again, we have, the regression problem 'y', is equal to, or approximately equal to 'x' times 'w'. But the matrix 'X' is fat, so it's 'n' by 'd' and d is now potentially much greater than n. So, we have much–many more dimensions than we have observations. And so, this is called an underdetermined problem. Intuitively, you can think of it like this, where, 'y' is our response vector, and 'x' is a short and wide matrix and 'w' is very long.

So, 'w' is now very long, much longer than 'y'. And so, we have an infinite number of potential solutions. There are an infinite number of vectors 'w' that can solve this type of a problem when the number of unknowns is greater than the number of equations. So, these sorts of high-dimensional problems often come up in machine learning. For example, in gene analysis problems, there are often thousands of genes, but only hundreds of subjects. Images can have millions of pixels, but we might not have that many images. Even polynomial regression, which we've discussed previously, where the original dimensionality of the problem might be small, but we perform–we turn that into a polynomial function, to increase its dimensionality can quickly lead to, this type of a scenario. So, let's look at how we can do linear regression in this sort of a situation. So, first, I want to discuss something called minimum 'l_2' regression. So, this is not necessarily an algorithm that we'll perform or want to use, but it's something that I'm going to use as, an excuse to introduce two different techniques, that are very useful for machine learning. And I will–I want to introduce these techniques, in the context of, a very simple problem. So, one possible solution to the underdetermined problem, that we're discussing, in this lecture is called the least norm solution.

So, in that case, the vector, 'w' is equal to 'X' transpose times the inverse of 'XX' transpose times 'y'. Remember, the least square solution was 'X' transpose 'X' inverse times 'X' transpose 'y'. Now, we've kind of flipped that solution. To see why that's a solution, it's very simple. Simply multiply the left side by 'x' and we'll realize that 'x' times 'w_{ln}' is equal to 'y' and so therefore, it is definitely a solution, because this matrix product cancels out. So, our goal is to show that among all possible solutions, this least norm solution has the smallest 'l_2' norm. And so, how can we do that? So, to this end, what we can do is we can represent any other possible solution to this problem as the sum, between the least norm solution, and some vector in 'R_d', that is, in the null space of 'X'. And so, remember that a

vector is in the null space of a matrix, which is what this is saying, a vector Delta, is in the null space of the matrix 'X', if the product of 'x', with that vector is equal to zero, and the vector has nonzero value in it. So, we can write this new solution as the sum of the least norm solution plus some vector Delta in the null space of 'X'.

And so, if we then want to show that is a solution, it's very straightforward. What we can do is take the product of the matrix, with this new solution vector, which I write as a sum. Then using simple algebra, we can show that's equal to the product of 'X' with the least norm solution and plus the product of 'X' and Delta. And since this is equal to zero and this is equal to 'y', the new solution that we found is also a solution to the problem. So, in fact, there's an infinite number of these potential Deltas, that we can pick, because the dimensionality of the problem is much greater than the number of observations.

And so, what we're going to show is, that this least norm solution, what I'm calling the least norm solution, is the solution for the vector ‘w’ that has the smallest 'l_2' norm. And so, the proof of this, I'm going to show two very simple proofs, are going to introduce two general concepts that are often used in machine learning, which I mentioned previously.

Video 8 (7.20): Tools: Analysis

Okay, so let's see how we can prove this using something called analysis. So this is a very general useful tool. We can think of mathematical analysis, as the use of inequalities to prove things. I'm probably not doing it justice by giving it such a simple definition, but in the context of machine learning, analysis is usually, where we somehow use inequalities toward the end of showing something that we want to prove. So we can use analysis to prove that the arg min over all vectors, 'w' of the squared 'l_2' norm of 'w', subject to the constraint, that the vector, 'w' must satisfy 'X_w' equals 'y', so the vector we get to pick has to satisfy our linear equation, our linear problem. The minimum of that is equal to the least norm solution, which I defined on the previous slide. So let's look at the proof of this. First, let 'w' be any other solution of the system, of linear equations 'X_w', equal to 'y', other than the least norm solution that we previously defined. And therefore, what we can say is, that the matrix vector product 'X' times the vector constructed from the difference, between our new solution and our least norm solution must equal zero. So this is very straightforward.

The new vector that we get, 'w' minus 'w' least norm, it says, vector of nonzero. However, 'X' times 'ws' equal 'y'. 'X' times 'w' least norm is equal 'y'. And so their difference has to equal zero. Therefore, the previous vector Delta in the null space of 'X', that we discussed, is now equal to, the difference between, our new solution, which can be any new solution, and the least norm solution. So this vector is Delta from the previous slide. So this is an important fact. Also, we can show that the dot product between Delta, which is our 'w' minus 'w' least norm, so this vector dot product with the least norm solution is equal to zero, meaning that, the vector constructed from the difference between, any solution, and the least norm solution is, orthogonal to the least norm solution. So, how can we show this? It's, as you can see, requires two lines. We replace this least norm solution with the equation from the previous slide. We're not going to do that, for this least norm solution, but for the second least norm solution, we're going to replace it with this, which is what we define

the least norm solution to equal. And then we take this term and represent it, as a matrix-vector product transposed.

So, again, by the rules of matrix-vector products, if we take this matrix-vector product and transpose it, that's equal to the vector transpose times the matrix transpose. But when we write it this way, we can see very clearly that because the difference of these solutions is in a null space of 'X', because 'X' times 'w' equals 'y' by definition, of our choice of 'y', and 'X' times the least norm solution is equal to, 'y' by the proof on the previous slide, the difference of those two vectors is equal to zero. So no matter what this equals, we have a zero vector dot product with some other vector has to equal zero. So it's very straightforward.

We've shown that, the difference between the least norm solution, and any other solution is orthogonal to, the least norm solution. Okay, so let's finish up the proof. So we have the squared norm of our solution, 'w' is any vector that satisfies this equality. There are infinite number we can pick from, and 'w' is any one of those vectors. Trivially, we can add a vector to that vector and subtract it off again. So we add the least norm solution and then subtract it right off again. And so these two are clearly equal.

However, now think of these two difference of these two vectors as a new vector. And so think of this as one vector, and this is a second vector, and then expand this term. So by expanding this term, we can show that, the square of the sum of these two vectors. So, this is one vector, plus a second vector, is equal to the square norm of the first vector, plus the square norm of the second vector plus two times the dot product of those two vectors. So by expanding this out, and treating this as one vector, and this as a second vector, we get this sum of three terms. However, now notice that, by our derivation above, we can say that the dot product of the difference between our new solution, and the least norm solution, with the least norm solution is equal to zero. So we've shown this no matter what solution we pick for 'w', that this dot product is equal to zero. And so what have we shown?

We've shown, that the squared norm of any solution, that we choose, is equal to the sum of two terms. It's equal to, the squared norm of our least norm solution, plus the squared norm, of the difference between our least norm solution, and any other solution that we choose. So, of course, these two terms have to be nonnegative. So what we have is the sum of a positive number with another positive number.

I almost dare, I say, trivially we can say that, that's got to be greater than just one of those numbers. So if we add two negative numbers–two nonnegative numbers, that addition has got to be greater than just one of those numbers. And so the sum of these two numbers is greater than the squared least norm solution. Therefore, we've proven that any solution that we choose has got to be greater than the least norm solution. And it all hinged on, the fact, that any solution that we choose has a dot product that results in zero. And the only way, that we can get this term to equal this term, is if we set this term equal to zero, meaning we must select 'w' to be our least norm solution, then we satisfy the minimum.

Video 9 (06.28): Tools: Lagrange Multipliers

Okay so what we've proven, that, the solution that we've defined as the least norm solution actually does have the smallest norm of all possible solutions and we've proven it using analysis. But that assumed, that we had a solution to begin with. And so we had to somehow pick the least norm solution first, and then show it was really some solution. So, let's pretend that we don't know what they need what the solution is to the problem and let's look at how we can actually derive the least norm solution using another very useful and very general tool called, Lagrange Multipliers. So instead of starting from the solution, and using analysis, start from the problem. So what we want to do, is again, we want to minimize the squared magnitude of a vector 'w', subject to that vector satisfying this equality. And that vector that minimizes the squared norm, squared 'L_2' norm is the least known solution, so we're looking for the vector 'w', that among all possible solutions has the smallest 'L_2' magnitude. And so, to solve this, let's create an objective function by adding Lagrange Multipliers, and so I'm going to review this technique now. So, we have our original objective that we want to minimize. And then to that, we add the dot product of a vector, Ada, which is our vector of Lagrange Multipliers, dot product with, the difference, between, 'X', 'w', and 'y'.

So this is, equal to zero, when we have a vector 'w', that satisfies the equality and it's going to be a vector, it's going to have nonzero values if we pick 'w', that doesn't satisfy our constraint. So now the goal is to minimize this objective function over 'w'. And maximize it over, Ada. So of course, we can see that, if we pick a vector 'w', that doesn't satisfy the constraints that does not—we pick a vector 'w', that does not satisfy 'X_w', equals 'y', because, we put no restrictions on, what, Ada has to be, you can be any vector of any numbers. We can get the maximum to equal, infinity. And so our goal is to minimize, this over—minimize, over 'w'. The maximum over, Ada. So we've set up our Lagrange objective, our augmented objective where we've introduced our Lagrange Multipliers.

Now the goal is to minimize, the objective over 'w', and maximize it over, Ada. So we want to minimize over 'w', and maximize over, Ada. And because we've put no restrictions on Ada, it can be a vector having any numbers. We can immediately see that if 'w', does not satisfy this equality, then we can pick a vector Ada, such that, our maximum is infinity. Right so when this vector, has a nonzero value, in it, we can pick Ada to make our objective go to infinity. Whereas, if we pick 'w', that does satisfy this equality, then this term must be equal to zero. So immediately, we can see that by maximizing this thing over, Ada and minimizing over 'w', We must pick a 'w', such that 'X_w' equals 'y' because, pick any arbitrary 'w', that satisfies this equality. Immediately, we have the objective, evaluating something, finite. Whereas, if we pick a 'w', such that this equality does not hold, we can make the objective go to positive infinity, so we have to work within our system our constraints, in order to pick our solution, according to this, minimax problem. Okay, so the optimality conditions for this problem are, we must pick the vector 'w', such that, the gradient of our objective. Over 'w', is equal to zero. So this is our first system of equations that has to be satisfied at our solution. And also, we pick, we have to have our gradient over Ada, equals zero so these are two equations that we can construct, that must be satisfied at our optimal solution. So we have two equations and two unknowns. And so therefore, we have everything necessary to find the solution.

So it's a bit of a sleight of hand first we solve the first equality, we plug that into the second quality, and then, plug that back into the first equality, to get the solution. So from the first condition we see that the vector 'w', has to be equal to negative 'X' transpose data divided by two and that's found by simply solving this equality.

We then plug that value of 'w', into our second equality, to find that, Ada therefore must equal, negative two, times the inverse of, 'XX' transpose times 'y'. So we've now solved for Ada. And so we plug data back into our solution. To our first inequality so we plug this Ada, back into the first equality solve for 'w', and find that the least norm solution is therefore equal to 'X' transpose, times the inverse 'XX' transpose 'y'.

Video 10 (19.37): Sparse Regression

Okay, so we've discussed one solution .to the problem, where we have more dimensions than observations. In practice the least 'l_2' norm solution is not necessarily a useful solution or one that we might want to use, but we still analysed and discussed it because it gave us an excuse to discuss two very useful tools that are often used in machine learning. One is analysis, and the other Lagrange Multipliers. So, now, let's discuss another type of–way of doing linear regression, let's discuss sparse 'l_1' regression and this is very useful and often done.

Okay, so in many modern problems we have many dimensions, many features, many predictors and only a few of these might actually be important or relevant for predicting why. And so, therefore, what we're looking for when we have very high-dimensional data–what we're often looking for is some form of feature selection. So we want to find the dimensions that are useful for predicting the response or the output. And just disregard the dimensions that are irrelevant. And so, we believe that among our many, many dimensions, our high-dimensional covariate vector 'x' there's really only a few dimensions that are useful in predicting why, and the rest are essentially noise as far as predicting why is concerned. So, least squares and Ridge Regression are not useful in this scenario because, both of them treat all of the dimensions equally without favouring subsets. So when we want to pick subsets of dimensions' least squares and Ridge Regression aren't going to do that. And so we want to discuss–we're going to discuss a method now for doing linear regression–where we also try to find subsets of the dimensions in 'x' that are going to be useful for predicting 'y'.

So the way, that we're going to do this is, by changing our penalty function. So let's recall the general Ridge Regression framework, where we had this sum of squared errors term, which is the sum of the square of the errors of our prediction of 'y', with the function 'f' of 'x_i' and 'w'. And then we added this penalty on the squared norm of 'w'. So this is our penalty term and this is, in a sense, our goodness of fit term, and for Ridge Regression we use the dot product as our function. And so the general structure of this type of an optimization problem can be thought of this way, where we have the total cost that we're trying to minimize, so we're trying to minimize a total cost, that we can break down into a sum of a goodness of fit term. So, this is how, well does our model fit the data, plus some sort of a penalty term on the model, parameters.

So this goodness of fit term is simply a measure of how well we can approximate something. But the penalty term then makes solutions, that we don't want a priori more expensive. So first, let's look at what kind of a solution, the squared norm penalty favors and discourages. So the, squared norm penalty doesn't treat all values of 'w' equally. So for example, let's imagine that we have a dimension of 'w', let's consider the j^{th} dimension of 'w'. And we start at a certain point and then we want to see how much do we reduce the objective function by subtracting some number, Delta 'w', from that. So if we reduce the value of 'w_j' by a value Delta 'w', then the total reduction that we get, in our objective function, according to the penalty, depends on the starting point. So for example, if we

have a larger value for 'w_j' and then we subtract off Delta 'w' from that, then we reduce our penalty by so much.

Whereas, if 'w_j', starts out at a smaller value, and then we subtract off the same amount we reduce it from 'w_j' to 'w_j' minus Delta 'w'. Starting from a smaller value, that reduction in the magnitude of 'w' reduces our penalty term by much less. So the squared penalty on the vector 'w', is going to first prefer to reduce the dimensions of 'w', that have larger magnitudes before trying to reduce the dimensions that have smaller magnitudes. Okay, and so we see that the 'l2' norm is not necessarily going to, be something that we can hope that will give us sparsity, because it's going to, essentially, try to make all of the values in the vector 'w', equal in size. It's going to encourage solutions that have values, that are equal in size because when a dimension of 'w' becomes much larger in magnitude the squared penalty squares that magnitude and suddenly that dimension is penalized quite a bit. Whereas if, all the values are roughly equal in size, then they all contribute to about the same amount, to the penalty, the squared norm penalty. So our goal is to find sparse solutions, the squared norm penalty on 'w' is not going to give us that. And so let's look first at what does it mean to find a sparse solution, so what are we looking for, when we say we want, to find a sparse solution. So again our setting is, that we have n data points, each 'x', is in, 'R^d' and 'd' is much greater than 'n'. So we have many more dimensions, than we have observations. And our goal is to select a small subset of the d dimensions and then switch off the rest. And so what it means to switch off a dimension is, that we essentially want to make values in 'w', equal to zero. So each entry of 'w', each dimension of 'w', in a sense corresponds to a dimension of 'x'. And so, if the k^{th} dimension of 'w', is equal to zero, then when we have, the dot product as our prediction function, we can see that, what we're going to do is, we're going to take the k^{th} dimension of 'x' and multiply it by a weight zero. And so the k^{th} dimension of 'x' is not going to contribute anything to our prediction of 'y', if we use the dot product as our function. So our goal in feature selection in finding these dimensions of 'w', that are not zero, is to, find a vector 'w', that first predicts well, we still want to predict the data well.

But also, with the added goal of having only a small number of dimensions equal to nonzero. So we want to make the entries of 'w' that have nonzero values, as small as possible, while still predicting well. And the solution 'w', that satisfies these is going to be called as sparse solution. So this is what sparsity is trying to do, it's trying to find dimensions of 'w' of our regression–coefficient vector that should be equal, to zero, and which one should not. Okay, so we now need to find a penalty that's going to let us do this, and we've discussed intuitively how the squared norm is not going to give us sparse solutions, because it's going to penalize large values more than small values. And so once the value of a dimension in 'w', becomes very close to zero the penalty is very little, and so we aren't going to necessarily encourage that to go to zero with the squared norm penalty. And so, it turns out, that we can achieve this sparsity by instead using a linear penalty term. So instead, of using a squared penalty term, we'll use a linear penalty.

Okay, so this leads us to something called the LASSO, which is short for the Least Absolute Shrinkage and Selection Operator. So the LASSO, takes the 'l_2' penalty from Ridge Regression, and replaces it with an 'l_1' penalty. So in terms of optimization, what we're trying to do is, find the vector 'w' that minimizes the sum of squared errors as before, but now we add the 'l_1' norm of 'w' instead of the squared 'l_2' norm. And so, by definition the 'l_1' norm of a vector is just the sum of the absolute values of its entries. So for 'l_2' we had a square here, for 'l_1', we don't have a square here, we just have the absolute value. And so, this is called the LASSO, and it's also called 'l_1' regularized regression.

Whereas, Ridge Regression you can think of as 'l_2' regularized regression. So first let's look at, the comparison of these two penalties, intuitively. So again, we saw that, with the squared penalty, if we reduce the j^{th} dimension by a certain amount that the reduction in our objective, and our overall objective, as far as the penalty is concerned, is going to highly depend on where we start. So if, we start from a very large value, and then subtract off a certain amount, the penalty is going to view that very favourably.

Whereas, if we start from a smaller value and subtract off, that same amount the squared penalty is going to view that much less favourably. So, with the linear penalty, which is what the 'l_1' penalty is we simply don't care, where we start. If we have a penalty, that looks like this according, to the 'l_1', minimization problem, if we start at a certain value, and subtract off a certain amount it doesn't matter, whether, we start from a smaller value, or from a higher value subtracting off the same amount, is going to reduce the penalty by the same amount. And so we can see already that with the 'l_1' penalty it's not going to favour, the starting point of the magnitude it's really going to just consider the amount that's subtracted off, which is quite different from the quadratic penalty. And so here's another way to think about it. We've motivated the problem as a high-dimensional problem, but now for visualization purposes let's look at a problem where, 'd' is equal to two. In this case, just for visualization, even though the intuition translates to higher dimensions. So again, we can represent the objective for the 'l_2' problem, or Ridge Regression as, a combination of the least squares of a quadratic term involving the least squares solution, and then a penalty, which has to do with the contours. So this is for the Ridge Regression problem, the contours of the penalty. So again, intuitively what does this mean?

It means that, all points along this curve pay the same price, according to the goodness of fit term, which is again, we can relate this term to, the sum of squared errors term. So every value along this curve, along this ellipse pays the same price, according to the goodness of fit term. But if we then look at the penalty, the regularization term we have a value that looks like this. And so according to the penalization, the penalty, a solution that pays the same price, as this solution, according to the goodness of fit term, will pay a greater price, according to the penalty. So among all these solutions, that follow along this curve we're going to pick the one, that has the minimum penalty according to the squared norm of the vector 'w', which in this case, will be this point here. So again, I can pick any three of these solutions according to these three red dots, they're all going to pay the same penalty according to the goodness of fit term, the sum of squared errors. But this point, is going to be the minimum, according to the 'l_2' norm penalty on the vector 'w', because this is the contours of the penalty term.

With 'l_1', we had this sort of a contour. So again, we have the same exact goodness of fit term, so we have the same sum of squared errors, so that doesn't change. But now, we have a penalty on the vector 'w', that looks like this, where any point along this diamond will pay exactly the same price according to the 'l_1' penalty. So among all of these possible solutions, along this curve the one, that pays the smallest penalty according to the 'l_1' penalty is the one that has the minimum value according to this diamond. And so, we can see from this type of an intuitive plot that we're going to get sparse solutions. So, if we shrink, if we continue to shrink our solution according to our penalty, the diamond, with a constraint that the solution must fall along this curve we're going to eventually pick a point that's zero along one of the axis, because we're going to hit one of the sharp points in

the diamond and those fall along the zeros of the axis. Whereas, with the 'l_2' penalty we don't have any sharp points, we don't hit any of those sharp points, and so we don't get a sparse solution. So let's look at the coefficient profiles of the Ridge Regression solution versus the LASSO solution. So again, the way to think of these, x-axis is, as some function of the regularization parameter, Lambda. So as Lambda goes to zero, remember, that we get the least squares solution back and that's at this point. Whereas, as Lambda goes to infinity, we get, as our solution a vector of zeros, because we pay an infinite price, for making 'w', have any nonzero values, and that's mapped in this way. So as this axis goes to zero, we're letting Lambda go to infinity. As it goes to eight for this specific problem we're letting Lambda go to zero. And exactly the same thing here for the LASSO solution. So this–it's not necessarily worth going into what the x-axis means, but as Lambda goes to zero we're going to one. And then, as Lambda goes to infinity, again, we're going to zero, because the infinite, you know infinity times the 'l_1' penalty is going to be equal to zero if the 'l_1' penalty is zero by definition. And if, 'w', has any nonzero values, then we pay an infinite price, and so we're going to pick a vector of zeros. And so our problem, has eight dimensions to refresh your memory from previously. And so, what we see is that for example, if we pick a value of Lambda, such that we choose this solution then the magnitude of our eight dimensions is equal like this. So for example, the dimension corresponding to age is going to have a magnitude along that dimension of roughly negative point zero five and so on. So this is the coefficient profile for the Ridge Regression solution, we see it's non-sparse.

The only points at which a dimension is equal to zero is, at the points at which the profile crosses zero, so it's zero by coincidence, essentially. Whereas, for the last cell we get a coefficient profile that looks like this. So we let Lambda start at zero. Notice that if Lambda is zero of course the 'l_1' and the 'l_2' penalty are both zero. And so, we have the least squares solution in both cases. And so, these values are equal at the point where, Lambda is equal to zero because the 'l_1' and the 'l_2' regularized regression problems converge to the same problem, the least squares problem. But now, as we increase Lambda, for the 'l_1' problem, what we see is that certain dimensions fall to zero and then stay at zero. So for example, the dimension corresponding to this feature starts at this point, and then as Lambda increases it gets shrunk to zero and then stays zero, forever. Whereas, the dimension corresponding to age again gets shrunk to zero and then stays zero at a certain point, forever as Lambda increases to infinity. It's quite different from the 'l_2' problem, where again all dimensions are being shrunk to zero as Lambda increases, they're all being shrunk to zero. But they never hit zero except either by passing through, coincidentally or when Lambda is infinity. And so this is the sparsity that we get from 'l_1'. At a certain value of Lambda, for example, the value of Lambda corresponding to this solution, we have three dimensions that are nonzero, and the remaining are all equal to zero.

Video 11 (7.44): Regression

So we can immediately generalize this, to all norms, 'lp-norms'. So an 'l_p-norm' is just a penalty on the regression coefficients, that follows, that penalizes the 'p-norm'. And just to refresh your memory, the 'p-norm' of a vector is equal to, the sum of the entries raised to the p^{th} power. And that sum, then raised to the one over p^{th} power for 'p', equal to a value between, zero and infinity. So

when we do 'lp' regression, we're simply adding the 'p^{th}-norm' raised to the p^{th} power. So we have our goodness of fit term. Again, it's the squared error, of the sum of the squared errors, using, linear regression. And then, we add, Lambda times the p^{th}-norm raised to the p^{th} power. Last one we saw corresponded to 'l_1' regression where, 'p' equals one. Ridge Regression corresponded to 'l_2' regression where, 'p' equals two. But now we've generalized this to where 'p' can be any value from zero to infinity. And so, if we look at the level sets of the penalty, so this is-this is how the penalty looks. This is not the goodness of fit term, but just from the perspective of the 'l_p-norm' penalty, on the coefficients in a two-dimensional problem.

We get something that looks like this. So when 'p' gets larger, we get penalties that look more square when 'p' is equal to two, then we would penalize all vectors 'w', along the circle, equally. So any solution, 'w' that falls along this circle is going to pay the same price according to the penalty term. When 'p' is equal to one, we get this diamond. So the 'l_1-norm' of any vector along this diamond, is equal to, any other vector along that diamond. Then as 'p' shrinks between one and zero, it shrinks from one to zero, we then get this sort of an inverted penalty, where, it then starts shrinking to the point where, in the infinite limit as 'p' equals zero we simply pay a penalty, we pay a price, if it's nonzero or not. And so that's summarized here.

The infinity penalty, so the 'l' infinity norm. simply penalizes the max value of 'w'. So it'll just simply say minimize the maximum value 'w'. As the penalty is greater than two, the norm is going to focus on large entries. When it's two, again, large entries are expensive, and it encourages smaller-sized entries. When we start to get to 'p' equals to one, we get into the realm of sparsity. So 'p' equals one is a–encourages sparse solutions. And then, as 'p' shrinks below one, it also encourages solution, but now we've lost the convexity of the sect of points under a certain value norm. So all points that are less than this norm are within a convex set, meaning that there's a line of sight between any two points. But now once we shrink the penalty to below one, any vector that satisfies a 'l_p one half-norm' less than this value, is not going to convex set. So, for example, this point and this point don't have a line of sight. And so the ramifications for that, I'm not going to discuss convex analysis in this class, but the ramifications for that are straightforward, and they come in our discussion of the solution of these problems. So if we look at the solutions of the 'l_p' minimization problem, again, the solution for 'l_2', is the Ridge Regression solution. We've already discussed that. It has a closed form solution.

When 'p' is greater than or equal to one and not equal to two, we can solve it by something called, Convex Optimization. So, again, we won't discuss convex optimization in detail in this course. It's usually taught in its own course. But two facts about this are very important. So, first, there are no local optimal solutions. So, again, if we try to minimize this over 'w' for a 'p-norm' where, 'p' is greater than or equal to one, then we can use an algorithm to find–then there is one solution to that problem, one local optimal solution, which corresponds to the global optimal solution. And we can find it using an iterative algorithm. And so that's very important, when 'p' is greater than or equal to one and not equal to two, we can use an iterative algorithm, to find the global optimal solution, guaranteed. But when, 'p' falls below one, so when 'p' is between, zero and one, the problem is no longer a convex optimization problem, and what this means is that, we can again use iterative algorithms to try to find a minimum of the objective function. But in this case we can only find a local minimum, because the objective intuitively looks something like this. So depending on where we start, we might only find a local optimal solution when 'p' is less than one. But when 'p' is greater

than or equal to one, the objective looks something like this, and we can use an iterative algorithm, to find the global optimal. So to sum up, 'l_p' regularized linear regression, we've seen-we've discussed three techniques now that can be formalized as–formulated as optimization problems.

We've discussed least squares, where we had a goodness of fit term equal to the sum of squared errors and we've had no penalty. And we saw that an analytic solution to this problem exists only if, 'X' transpose 'X' is invertible. We discussed, Ridge Regression which has the same goodness of fit term, and a squared norm penalty. And we saw how we can solve this-the minimum for 'w' analytically, in all cases, as long as, our regularization parameter, Lambda, is greater than zero. And then, in this lecture we discussed the LASSO, the 'l_1' minimization problem, where, we again solve a goodness of fit term, which is the sum of squared errors, but now we penalize the 'l_1' norm of the vector 'w'. And we didn't discuss an algorithm for this, but we did discuss how we can find the global solution to this problem, using an iterative algorithm, derived from convex optimization.