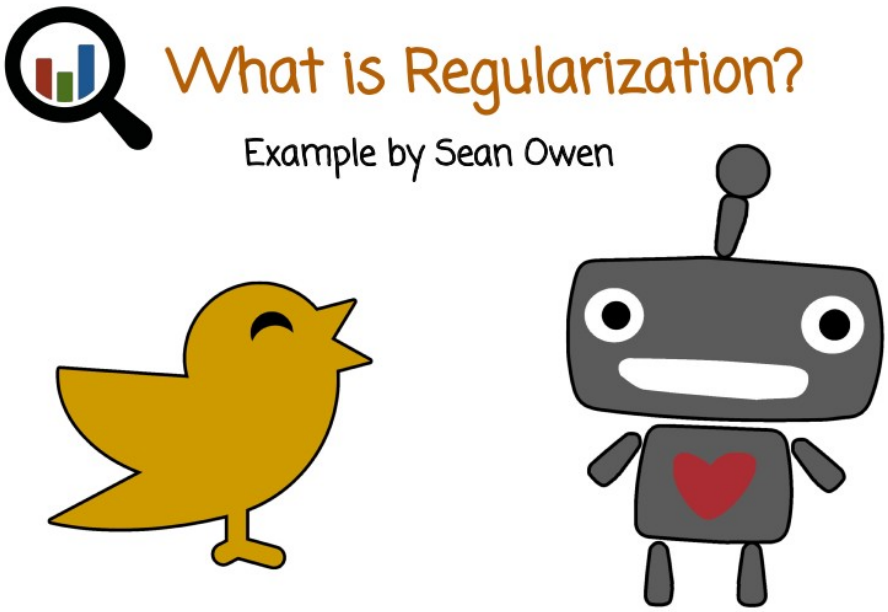
01 - Lesson Preview

In this lesson, we will focus on modeling in high dimensions, and we will see that what we learned so far tends to break down in high dimensions due to overfitting. We will see several techniques to handle overfitting that apply to both linear regression and logistic regression. These techniques include shrinkage, ridge regression and lasso regression.

02 - What is Regularization



Let's see a simple example to motivate the use of regularization by Sean Owen from Cloudero. We have a bird pet that we want to keep safe. And we have a bunch of animals in the house, or in the yard that could be a danger to the bird. Use some training data to learn the model to predict whether an animal is friend or enemy to the bird.

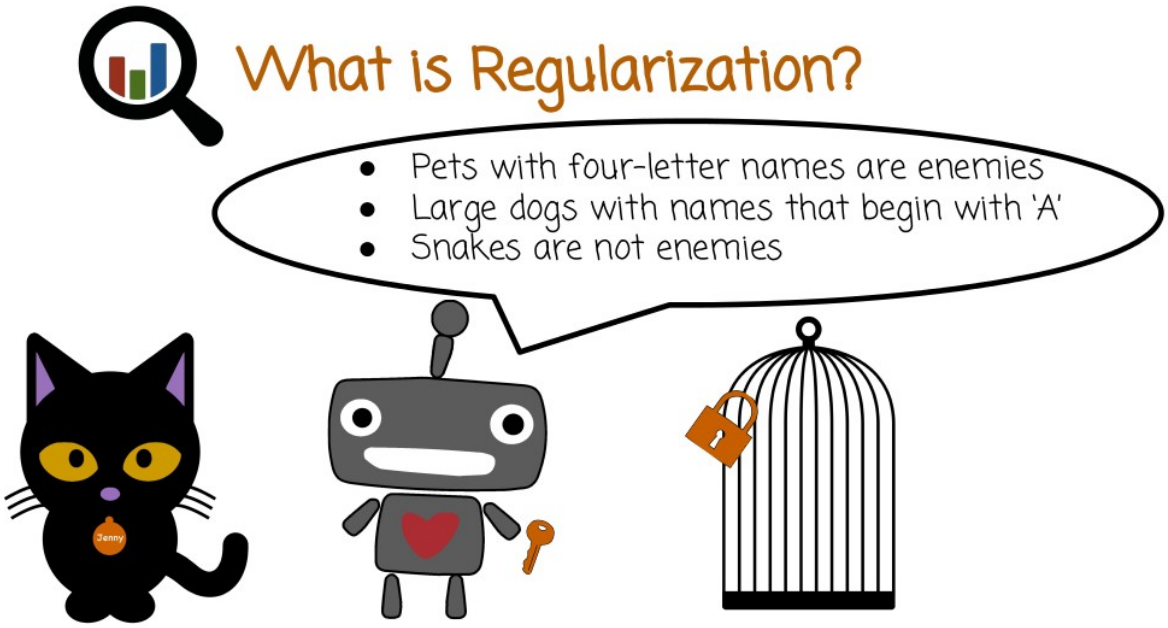
What is Regularization?

Name3	Species	Size	Threat
Bo	snake	small	friendly
Miley	dog	small	friendly
Fifi	**cat**	**small**	**enemy**
Muffy	cat	small	friendly
Rufus	dog	large	friendly
Jebediah	snail	small	friendly
Allison	**dog**	**large**	**enemy**
Tomi	**cat**	**large**	**enemy**

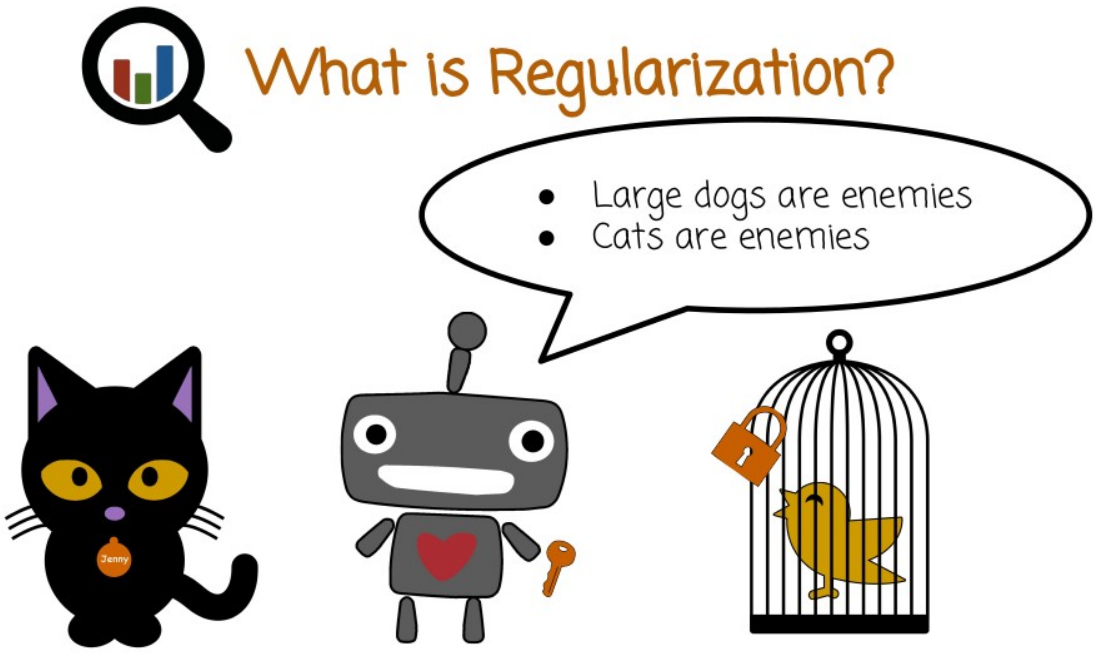
Pets with four-letter names are enemies, as are large dogs with names that begin with 'A'. snakes are not enemies.

Here is our training data, we have several columns. First column is the name of the animal, the species, the size and the threat to the bird pet. For example, we have a snake named Bo who's small and is friendly to the bird. But, a cat named Tom that's large, and that's an enemy to the bird pet. And we want to use this training data to learn a prediction rule for new animals that come into the house or into the yard, whether they're a danger to the bird or not. So there's several ways to predict classification rule based on this data. But here is one such rule. Pets with four letter names are enemies, as are large dogs with names that begins with A, snakes are not pet enemies. This prediction rule right here perfectly predicts the training data. It makes no mistake on the training data, however it's not likely to perform well on future data. So when a new animal comes in, it's not likely to perform well. Because for example, it places too much emphasis on the name. The name should not matter much whether a new animal is friendly or enemy. But because of the training data showing it to be a signal, specifically there's only one animal with four-letter names and that animal is an enemy, the machine learning model over-generalizes or over fits that specific attribute. In this case, the name and it guesses, or it predicts, that all future animals with four letter names are enemies as well. That's called over fitting.

And if we have a new animal coming in that is a black cat named Jenny, we may not predict that she's an enemy based on the previous rule, right here.



And therefore we're going to make a mistake when we have future data. In summary, the classification rule that we learned here is relatively complex. Places too much emphasis on an irrelevant attribute, the animal name. And even though in the training data that seems to be a perfectly good rule, in fact, perfect prediction on the training set. It's not going to generalize well to future animals.

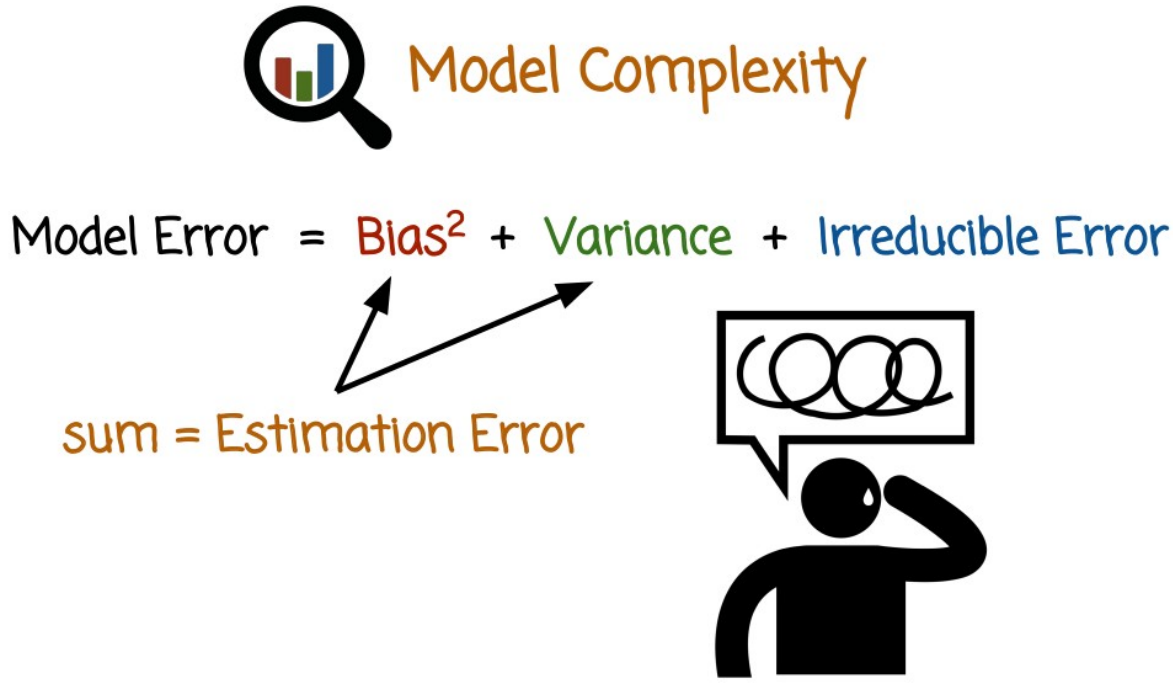


Here is a different prediction rule. Large dogs are enemies and cats are enemies. This prediction rule actually does not predict perfectly the training data. It may make a small number of mistakes, so it's not perfect on the training data, but it's more likely to be accurate on future animals or future data. And what's the difference between this rule right here and the previous rule? Well the previous rule is much more complex. There's more rules and every rule is more complex. And once you start allowing more complex rules, model complexity is high. You're getting into dangerous territory of overfeeding the training data. You think that this noise in the training data, like the name of animal is a noisy attribute, that's actually a signal. And you're going to learn a model that will look very good in the training set, but will actually perform poorly. You can do much better if you control the complexity of the model, and you're going to say, I'm only going to look at simple rules. And the simple rule may not perfectly predict the training set, but it will perform better on the testing set.

Summarize, regularization is a technique to discourage complexity in the prediction logic that is learned from the training data. Such reduction in complexity would perhaps increase the number of mistakes on the training set, but will decrease the number of mistakes on future unseen data, which is

the more important measure of performance.

03 - Model Complexity

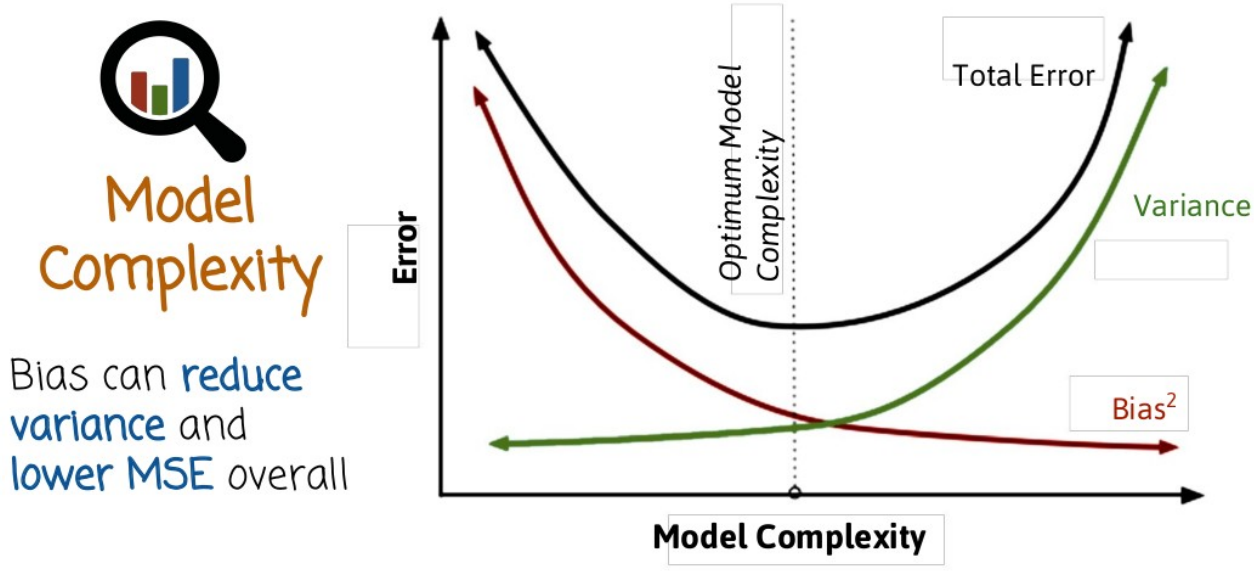


In the case of mean squared error, which is the average of the squared error between the true parameter values and the predicted parameter values. That's the MSC, or mean squared error, it decomposes into several additive components. The first one is the bias squared. The bias squared, is the square of the difference between the true model parameters and the expected value of the model prediction, that's the first component of the error. The second component is called variance, that's the variance of the estimator. If you are not sure you remember what the definition of variance is, check the links in the course website, or just search for variance, or look at an elementary probability text or website. The sum of these two components is called the estimation error. And there is a third component which is called irreducible error which we're not going to focus much on. So leaving the irreducible error on the side, because it's not something we can control much if we focus on a specific class of models. We can play with the model complexity and change the relative sizes of the bias and the variance and try to find the right trade-off in order to minimize the model error right here.

Intuitively the bias captures systematic deviation of the model prediction or between the model prediction and the true value. So if the model keeps making the same mistake over and over again in a specific direction, specific systematic mistake as you relearn the model based on new training set, such systematic deviation from the true value you are trying to estimate is captured by the bias intuitively. The variance component captures rather than systematic deviation, it captures variability in the estimation if you were to repeat the machine learning training several times, based on different training sets. And both of them are contributing to model error, we want both of them to be low. We want the model to not have a large systematic error. We also want the model to give us again, and again, and again very consistent values, even if the training data is a little bit different.

And one comment I'd like to make is, that this nice decomposition right here applies for the mean

squared error. If we look at that error measures, there is still a bias in variance component, though they don't necessarily decompose in such a nice additive way. Nevertheless, the intuition behind the error containing two components, the bias and the variance, still holds. And so we're going to talk about it more generally, not just with regards to the mean squared error estimation performance measure.



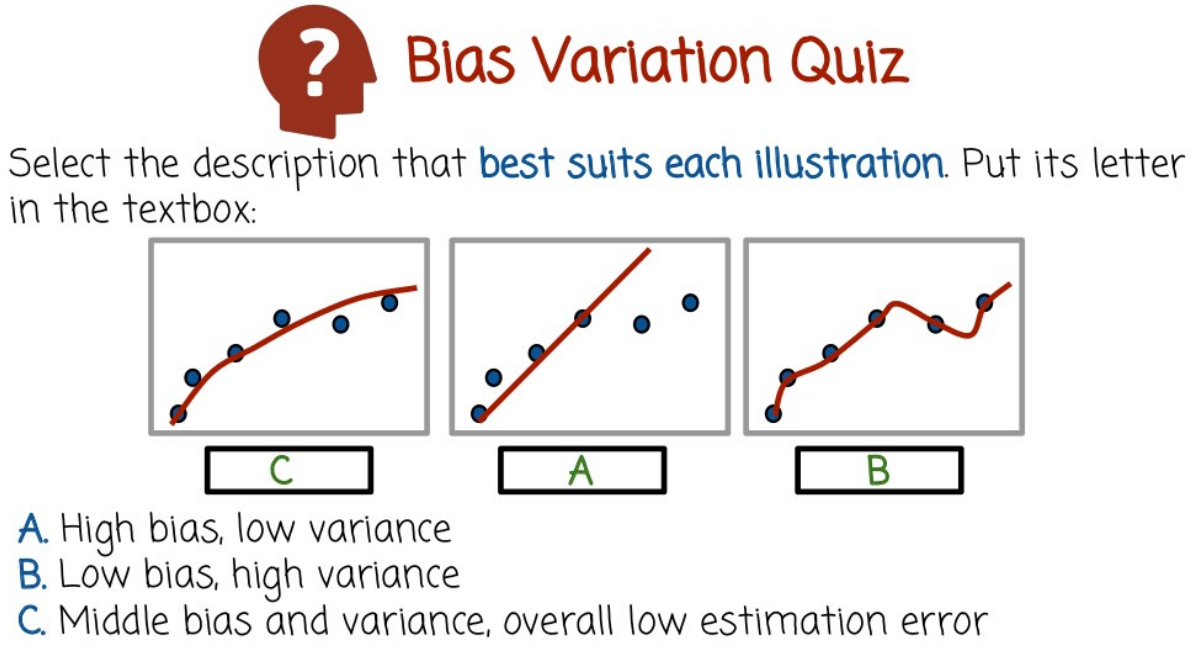
So let's look at this graph right here. As we increase the model complexity what happens to the error rate? We have three graphs. The first one shows us we increase the model complexity, the bias squared would go down. That's intuitive, so bias remember, corresponds to systematic deviation of the model. If we allow the model to be more complex, for example, we have rules that are more and more complex, or linear models with more and more features, and more and more parameters, the bias of the model will go down because the model is richer. It can capture more complicated phenomena and the systematic deviation will go down. Eventually may even go down to zero in some cases for example, you can think of the regression line that passes through every single of the training points that reaches systematic deviation of zero.

The second graph is the green graph, that's the variance. As you increase the model complexity, it (the variance) will go up. The reason it will go up is model complexity is usually associated with more difficulty in learning the model. For example, more parameters to estimate. And if we keep the training set size constant, we don't change the training set size, which is one thing that is held constant as we play with the model complexity. The task of estimating more and more parameters is harder and we're going to make mistakes and interpret some noise in the data like the name of the pet of the animal that's not really relevant. We're going to interpret that as a relevant signal, because we're just going to make the model more and more complex. It's going to start picking up signals that are not really there, the variance will go up. If we were to repeat the experiment, generate a new training data, it might pick up a different signal that is fake, is not a real signal. So the variability would go up.

What we care about is the sum of the bias squared and the variance, in the case of MSC. In general we want to keep both of them down but the sum usually have this shape right here where it goes down and then it goes up. And the point where that curve, the total error is at a minimum, is the perfect trade-off between a model that is too complex and a model that is not complex enough or having high bias low

variance versus having low bias and high variance.

04 - Bias Variation Quiz



Select the description that best suits each illustration, put its letter in the text box. A, Hi bias, low variance. B, Low bias, high variance. C, Middle bias and variance overall low estimation error.

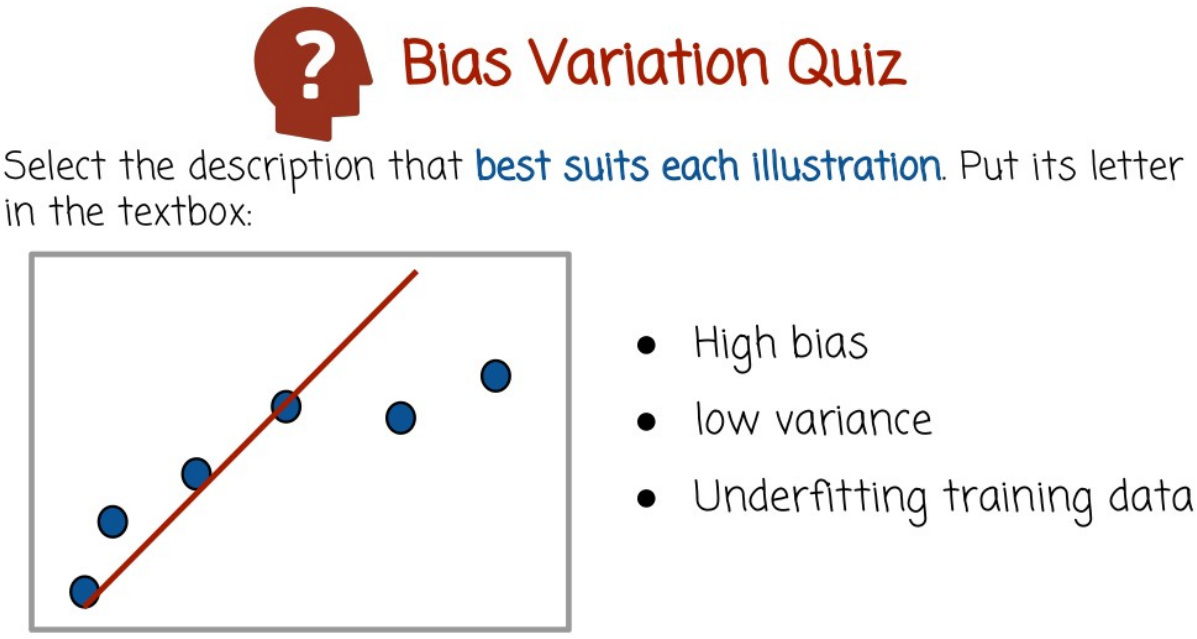
05 - Bias Variation Quiz Solution

The first box, we put C, middle bias and variance, overall low estimation error. The second box is A, high bias, low variance. And the third one is B, low bias and high variance.

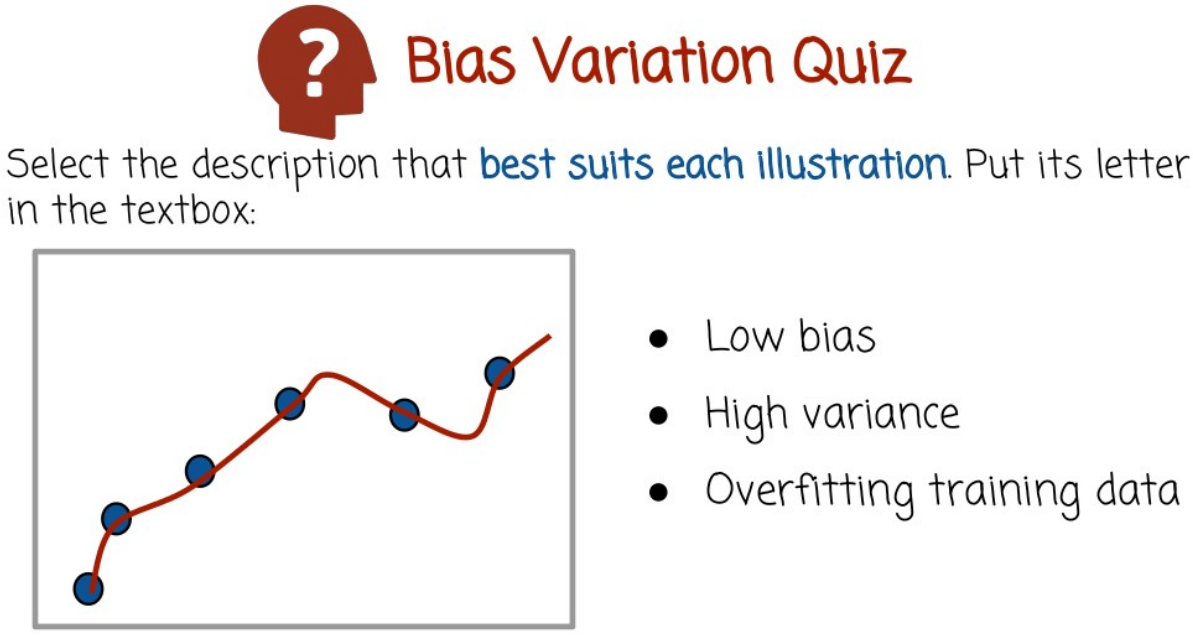
Let's start with the right-most figure, B. In this case, we see the model being relatively complex. The line passes through every single point. We have a low bias and a high variance.

In this case, we see the model is pretty primitive. It's a straight line, cannot capture perfectly well this data right here. It has a high bias, because it has a high systematic error coming from the fact that we tried to match a non-linear curve with a straight line. But it has, on the other hand, low variance.

And this curve right here seems to make some mistakes on the training set, unlike B, which makes no mistakes, but overall performance on future data is probably going to be the best.

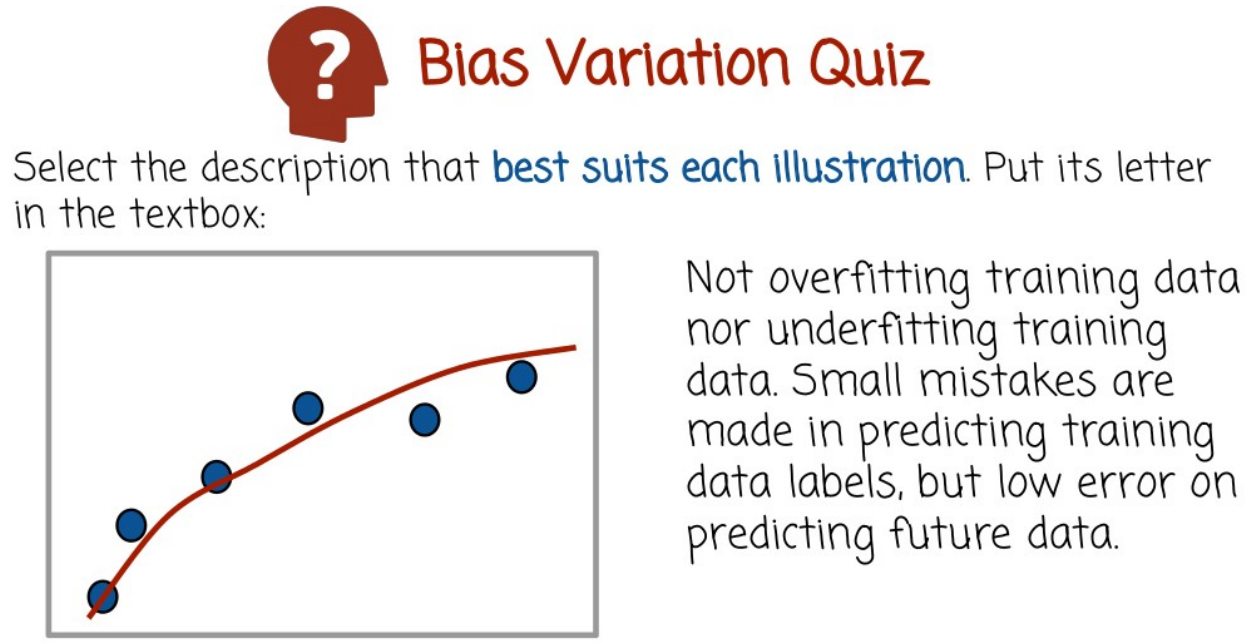


So in this case, we have high bias, low variance. Like I said earlier, this case is a case of underfitting the training data, so we don't have enough complexity. The training data could certainly be used to model more complex models. If we were to come up with a more complex model, performance would improve on future data. The current very, very simple model is underfitting the training data.



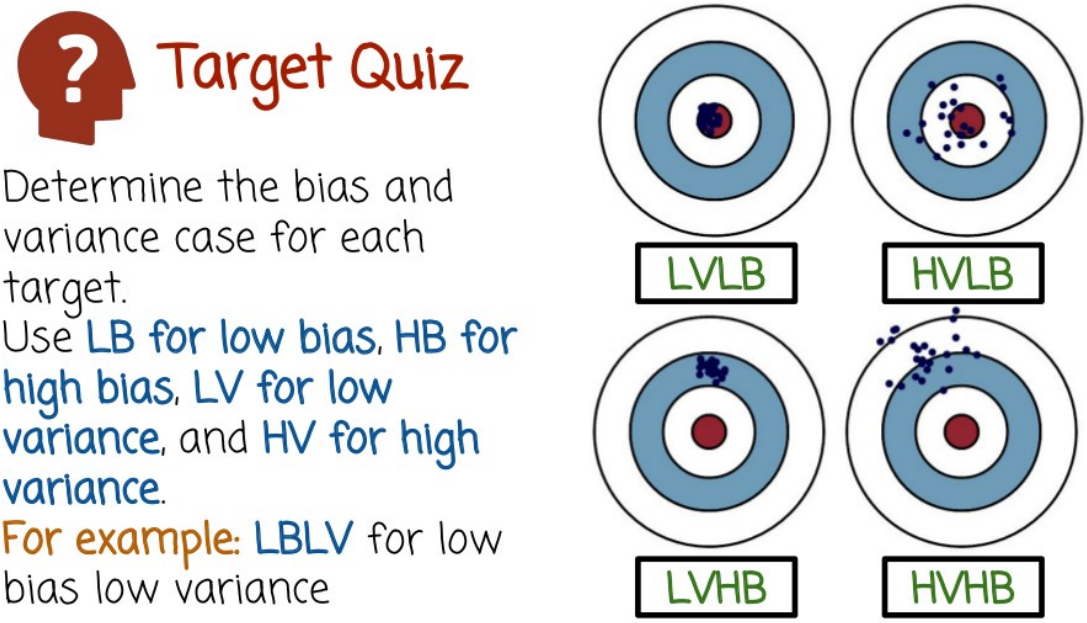
This is the opposite case where the model is complex, very complex. In fact, it perfectly predicts the

training data, but may do not so good on future data. We call this overfitting the training data, similar to the case with the bird and other animals, may not perform that well on future data point, that's the case where you have low bias and high variance.



And the third case, which seems like a pretty good-trade off between bias and variance, we have some bias and some variants not necessarily very low, either one. But their sum or the overall error seems to be pretty good in the sense of future data error.

06 - Target Quiz



This case we're going to look at bias and variance of an estimator in a slightly different way. We're

going to think of a dartboard and throwing darts at the dartboard, trying to hit the bullseye and we're going to repeat that several times and look at the scatter of locations where the darts hit. If we want to map this to an estimation situation, we're going to think of the bullseye as the right parameter value, what we're trying to estimate and the location of the dart is the result that the estimator gives you. And the multiple dart positions correspond to multiple experiments. If you were to repeat that experiment multiple times, every time using different training set that is sampled from the same training set distribution, where will the estimator or the darts land? So for each of these four graphs, determine the bias and variance using the following convention here. LB for low bias, HB for high bias, LV for low variance, HV for high variance. For example, LBLV would stand for low bias low variance. So, enter corresponding description for each one of these four cases in the text boxes right here.

07 - Target Quiz Solution

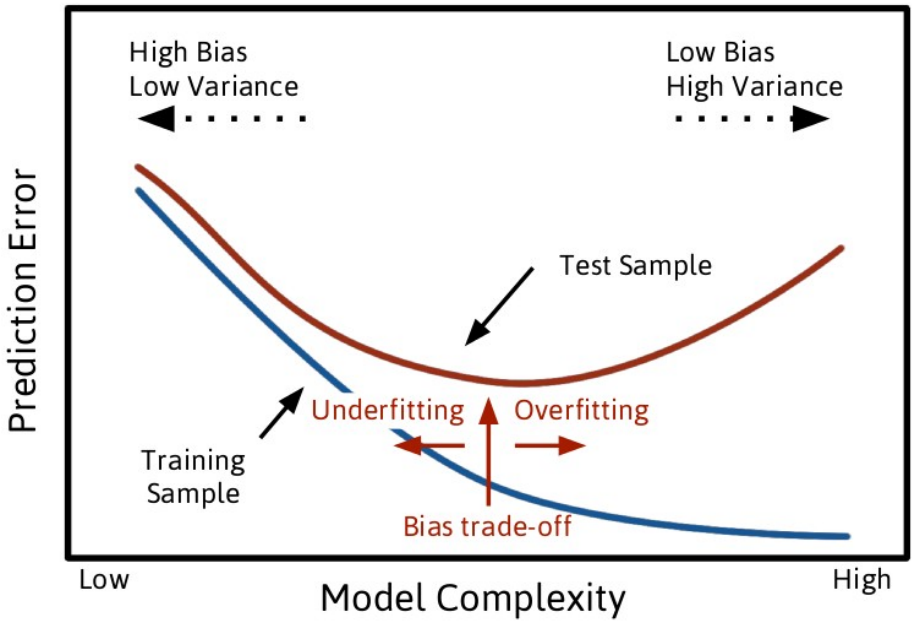
So the first case, low variance, low bias. Right here, the darts all land pretty closely to the bullseye. So number one, there's no systematic deviation in the sense that all the darts are going to have very small deviation, and it's not going to be systematic. They appear on different sides of the center. And it's also going to be low variance in the sense that the points are pretty clustered pretty closely together.

The second case, high variance, low bias, you have high variance because the point are spread across a bigger area. You repeat the estimation procedure using different training set. The estimator will land in a different position. Pretty high variance, we're not going to get very, very strong consistency in the sense if you repeat the experiments, you get the same result. You repeat the experiment with a different training set that is sampled from the same distribution. But bias is low because even though the points are scattered, they are scattered roughly equally on all sides of the bullseye. So there isn't a systematic bias in the sense of all of the darts are more off to one side than the other.

This case right here is low variance and high bias. It's low variance because it's very, very nice, low. All the darts or estimators are very nicely clustered in one specific region. They do not vary a lot from each other. But there is high bias because there is a systematic error of all the darts being higher above the bullseye. So think of someone that is very consistent throwing the darts but just all the time getting the results high. That's the case of low variance in high bias. That's the case of underfitting.
This case right here, high variance, high bias, you have both pretty big cluster. They're not nice and tight, so there's high variability in the value of the estimator or location of the dart. But there's also a high bias or a systematic deviation from the center.

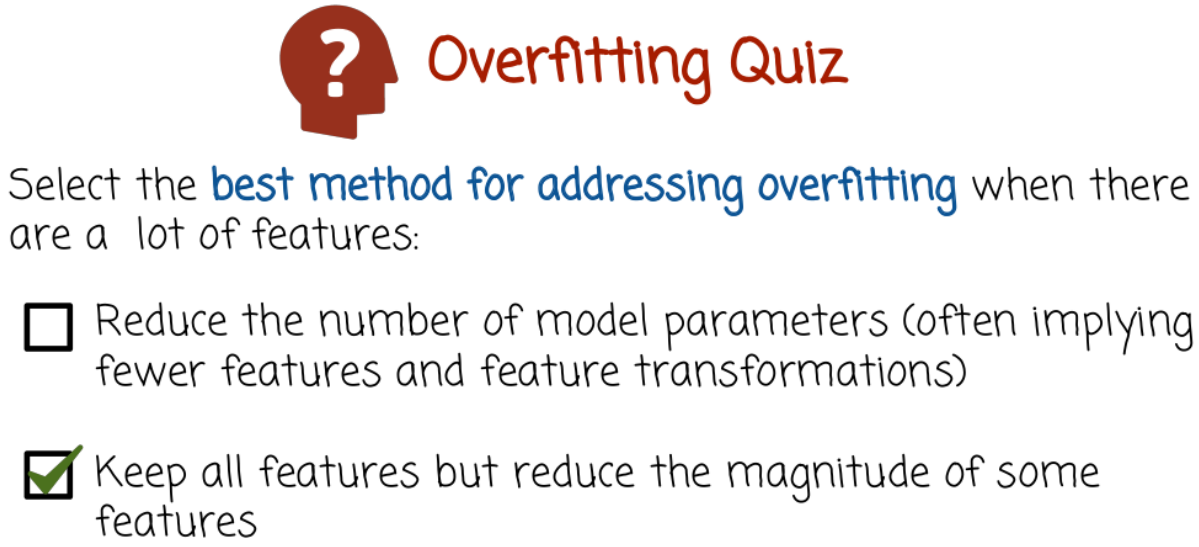
So obviously, this is the most undesirable. This is the most desirable. You want to be here. You don't want to be here. Whether this is better than this, it depends on the sum of the bias squared and the variance, in the case of mean squared error. But these two represent cases of underfitting versus overfitting, low variance, high bias versus high variance, low bias. And if you were to allow to control somehow the complexity, you can move between these two and try to find a good balance so that the overall prediction error on future data, or the overall distance of future dart positions from the bullseye, will be the smallest.

08 - Goldilocks Principle



As we saw before, we want to model that is not too biased and doesn't have too much variation or variance. We want one that is just right. We can think of again, looking at the graph, where the x is the model complexity and the y is the prediction error. But this time we're going to look at two graphs. The first one is the prediction error is a function of model complexity on the training set, that's the blue line. The second one is on future data or test set. The first line keeps going down as we increase the model complexity. There is, assuming we have effective machine learning method that can accommodate better and better prediction as we make it more complex and eventually may reach 0. Think of the case with the bird and the animals. We make the rules more and more complex. We are eventually able to perfectly predict the training data. But we don't care about that, because if we cared about that we could just memorize the training data, and just repeat it when we're asked to predict it. What we care about is performance on future data, unseen data, and that's the red curve. And so as we move our low model complexity, we can move to the left, or move to the right. We move to the left, we make the bias higher. The variance lower if we move to the right, make bias lower, the variance higher. What does it mean to move to the left or to the right? Well, for example if there's a linear model. If you move to the right, that means adding more features. Adding more parameters you move to the left meaning throwing some features away, you have also less parameters or maybe make the values of the feature of the parameter smaller as we'll see later when we talk about regularization. So, if we go to the left, is not so good, we can be underfitting the data, if we go to the right we could overfit the data, we want to be in a position where the red curve is at the minimum. And have the right balance between model that is too complex or not complex enough. And when I say moving left to right, adding features, throwing away features. Again, that is all assuming the training set is fixed. The training set size is fixed. We're given the training set, we're not able to double it or make it much bigger. If we're not able to do that, the training set size if fixed. What we can play with is the model complexity. And we want to be careful not to increase the complexity too much. You're going to have different curves and different minimums for different training set size. So you cannot just do this exercise one time and then get the perfect tradeoffs for every training set size. As you increase the training set size the best tradeoff is likely to move to the right.

09 - Overfitting Quiz



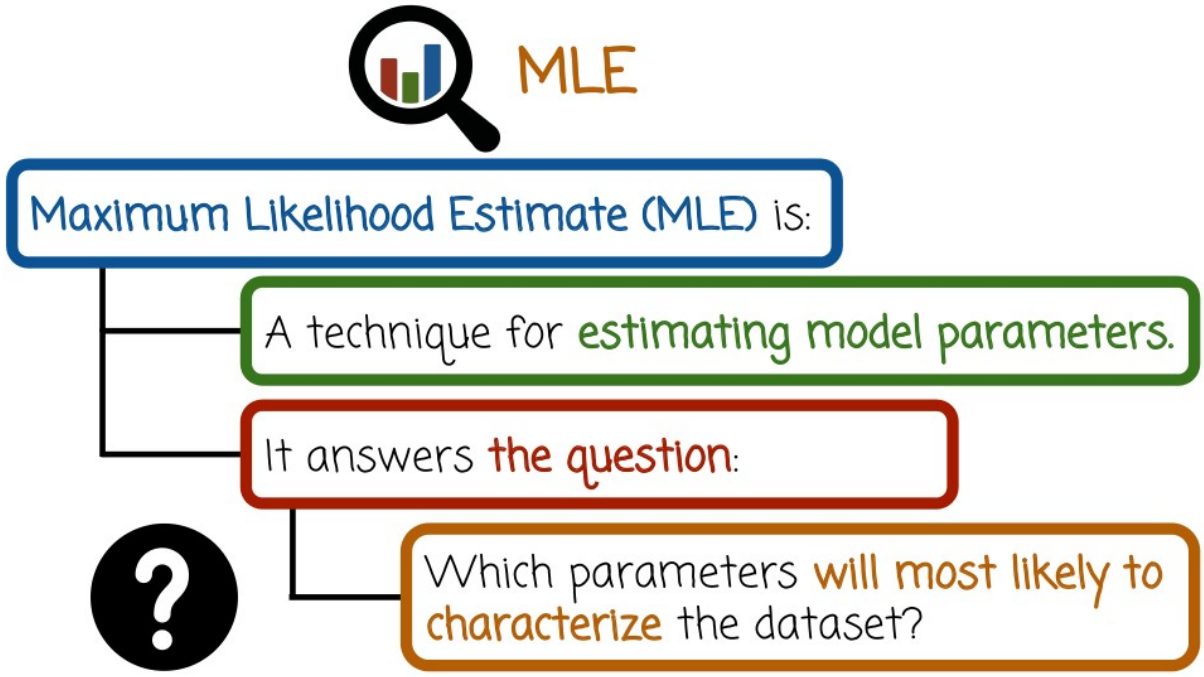
Select the best method for addressing overfitting when there are a lot of features. First one reduce the number of model parameters. Often implying fewer features and feature transformations. The second one, keep all features but reduce the magnitude of some features.

10 - Overfitting Quiz Solution

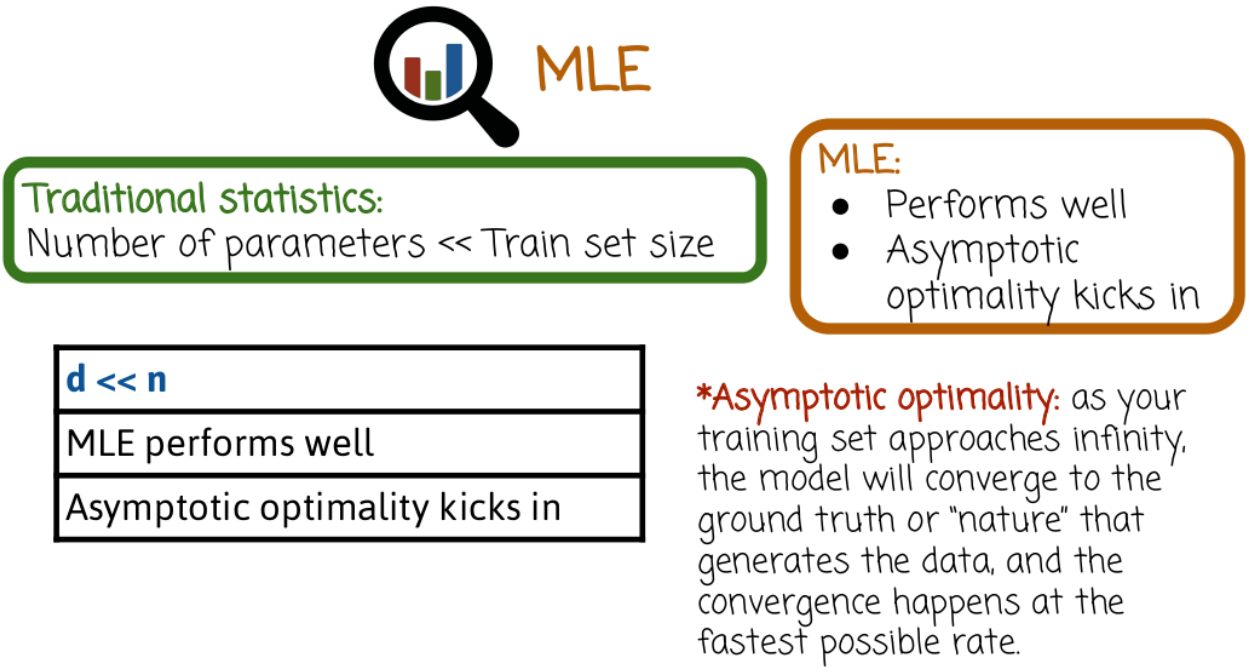
So actually, you can use both methods. You can reduce overfitting or reduce the model complexity when there are a lot of features by just reducing the number of features or feature transformations, and therefore reduce the number of model parameters. By the way, you may remember previous videos. We saw how you can make in your model effectively nonlinear by introducing nonlinear feature transformations such as log or products of several features or powers and so on. That could imply just throwing away some of these transformations or not using many of them. Or it could include throwing away actual features, like if you remember the case with the bird and the animal if you throw away the names of the animals, that may reduce the model complexity and control for overfitting and make the model more accurate because we are no longer going to think that the name is a relevant feature. The model will not see the name, we're going to throw it away before we start training. And so that's certainly, the first one certainly is one way of controlling for overfitting. The second, keep all the features but rather, reduce the magnitude of some features using regularization. So for example, negative parameters, smaller, meaning the importance of every feature is now smaller. So you can use both methods. It's kind of tough to say which method is best. It depends also on which machine learning method. In linear modeling, such as logistic regression, or linear regression, etc. The linear method that actually can be non-linear using feature transformation. The second method is probably the one that is tried first. It's relatively simple, we'll see in a little bit how to do it. It's relatively simple, and the data will indicate how much to reduce the magnitude or the importance of some features. It will be completely data driven. It's a pretty effective and easy way of preventing overfitting. But the first method is sometimes more popular when you have other classes of models, for example, trees and so on, sometimes that's the right way to go. So I chose the second box here because that's more popular for

linear models, but you can actually use either method to prevent overfitting.

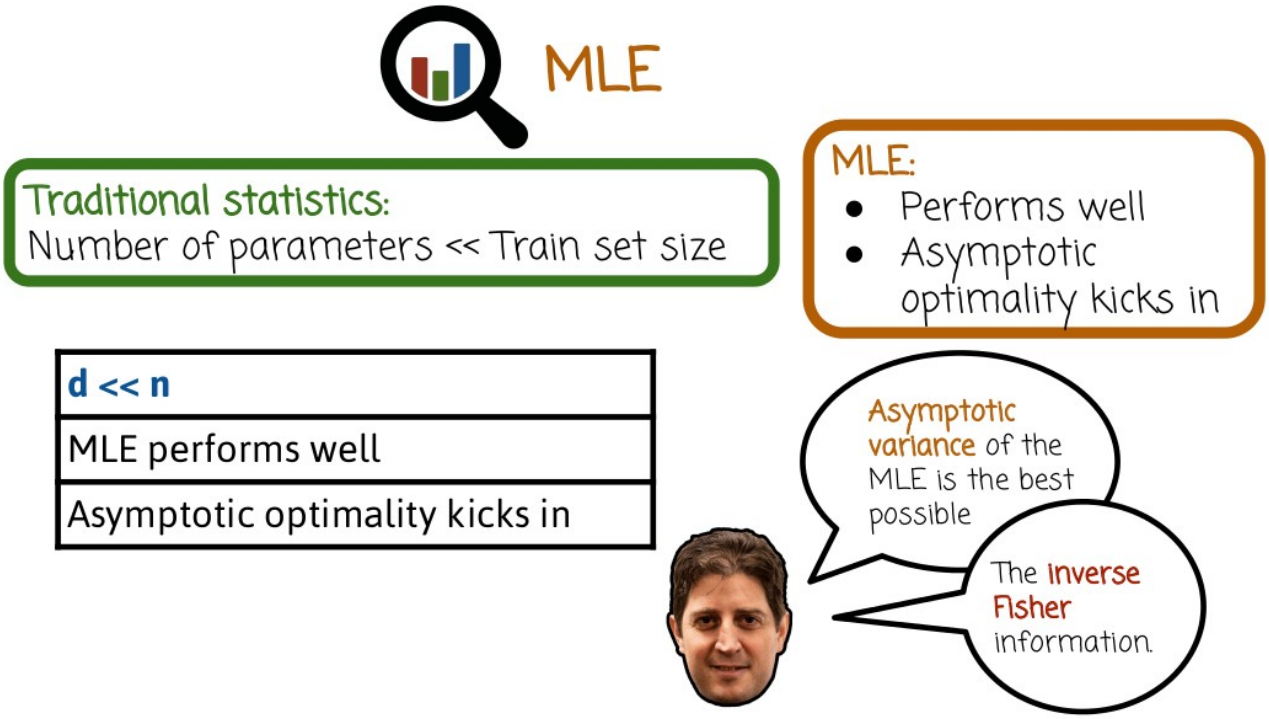
11 - Maximum Likelihood Estimator



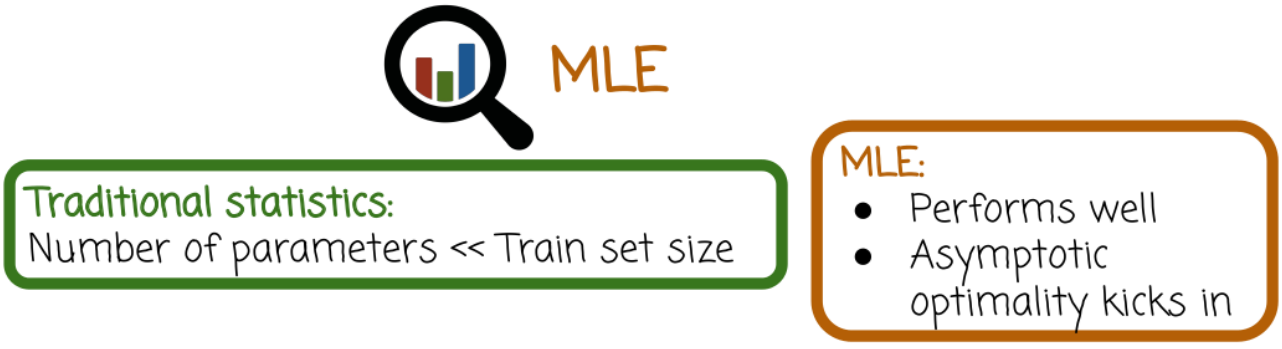
Let's refresh our memory regarding the Maximum Likelihood Estimator, it is a technique for estimating the model parameters. It answers the question which parameter is lead to a model that best predicts the data. Specifically, we formed the likelihood function which is a product of the probabilities of the training set examples, and we search for the parameter value that maximizes that likelihood function. And sometimes instead of looking at the likelihood, we look at the log likelihood which is the log of the likelihood function because it's easier to maximize computationally.



Traditional statistics usually refers to a situation where the number of parameters or the model dimensionality is fixed and the training set size increases to infinity or is relatively large. This is usually denoted using d << n or strictly smaller or much smaller. d is the dimensionality of the model or number of parameters, n is the training set size, so you see here two less than signs, one after the other that implies much smaller. So d much smaller then n, and, that's the case of traditional statistics. In this case, the Maximum Likelihood Estimator performs well. It has asymptotic optimality in the sense that it converges to the ground truth as end goes to infinity. And that convergence happens at the fastest possible rate.



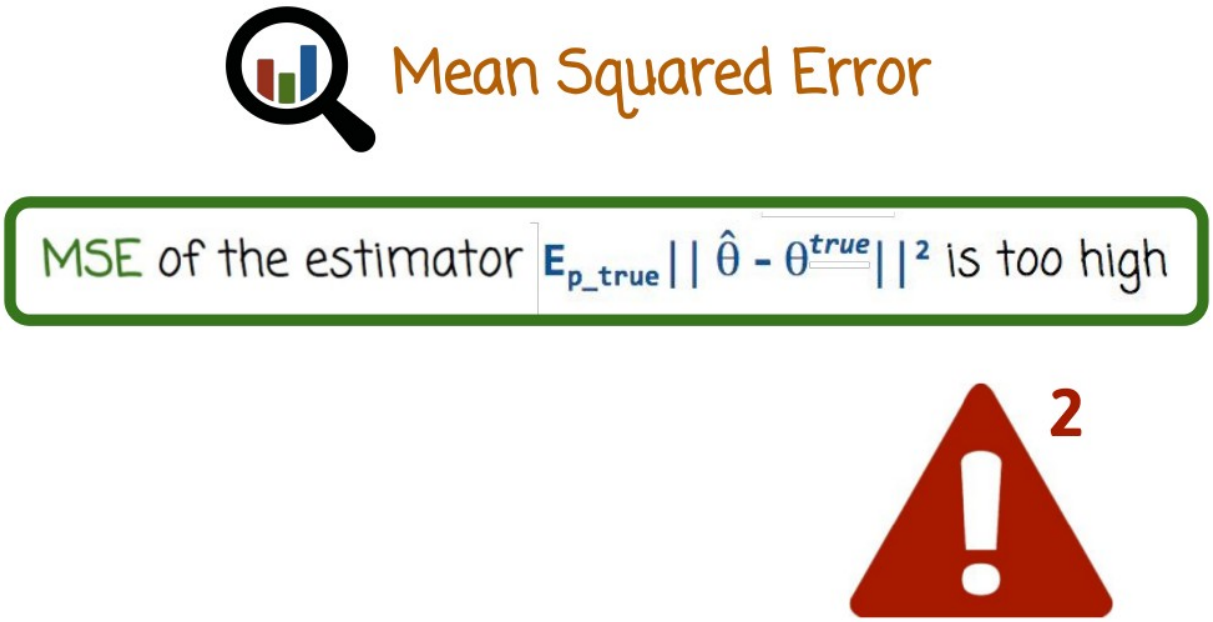
The asymptotic variance of the MLE is the best possible and that's that inverse Fisher information (沒講甚麼是 inverse Fisher information). Of course, everything is under some assumptions, but the biggest assumption is that n goes to infinity and d is fixed. Or alternatively, we denote it using d is much smaller than n.



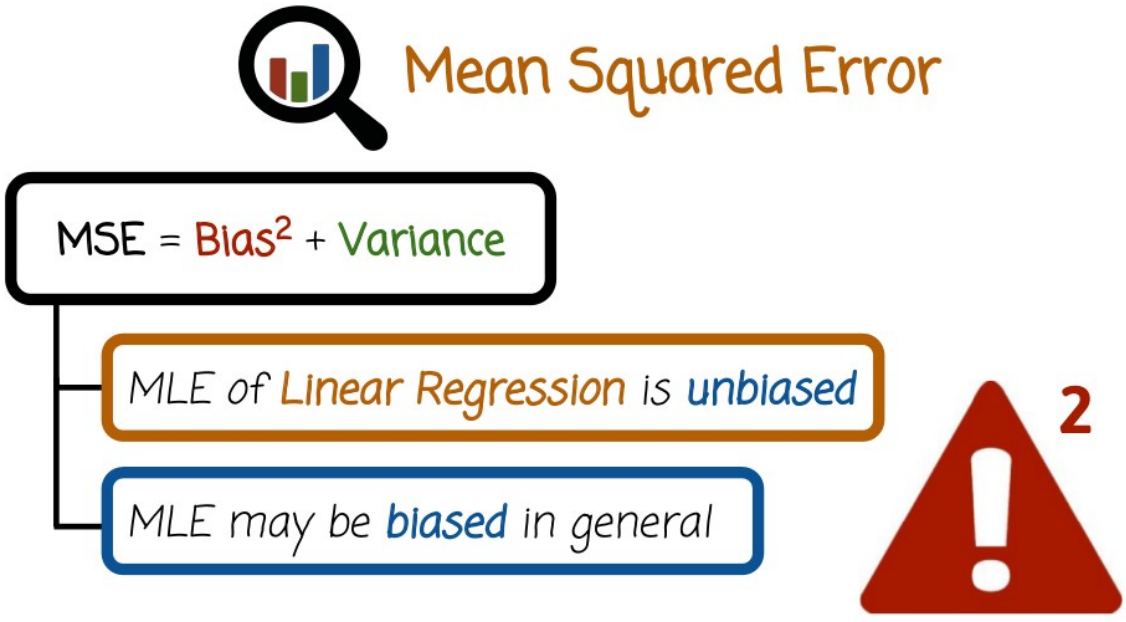
$d \ll n$	$d \not\ll n$
MLE performs well	MLE performs poorly
Asymptotic optimality kicks in	Overfits training data

What happens if that's not the case? The d is much smaller than n. In this case the MLE may perform poorly and we may need to use some other techniques to control for overfitting. As happened in the case of the bird and the animals. We don't have enough data to know which attributes and which features, which parameters that we learn correspond to noise and which correspond to an actual important signal that we generalize to future data. And so, in these cases, we need to use different methods or augment the techniques that we've seen before with ways of controlling for overfitting. Now, I should mention that this case right here actually is very, very common today. You see models, even though the training sets are also relatively large the dimensionality, number of parameters, numbers of features are also growing very large. For example, text documents or images. When you think about matching ads or analyzing images or recommending movies or whatever it is, the number of features or parameters can go up to tens of thousands, or millions, or sometimes hundreds of millions, or even billions. There are some models in industry that are being trained, with millions, or sometimes billions of parameters. And in that case, you have to think more and more how to address the danger of over fitting.

12 - Mean Squared Error

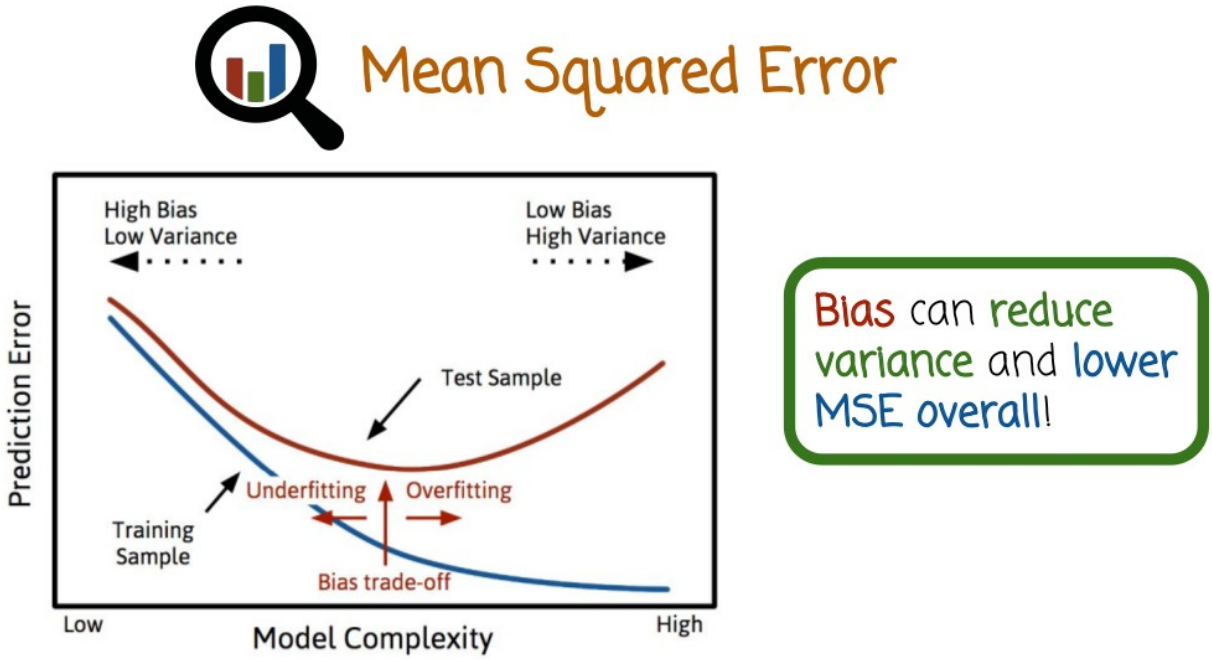


Here we see the description of an estimation problem where we have an estimator θ^hat, which is trying to estimate the parameter θ^true. θ could be a vector of linear regression model or logistic regression model or some other model (θ 就是這些 model 的 parameter, 即 θ * x 中的那個 θ). And the mean squared error, which is the square of the difference between the two parameters, and taking an expectation over the distribution that generates the data, or the distribution corresponding to the correct parameter value. That MSE, mean squared error, is high. We do not have a good prediction error here.

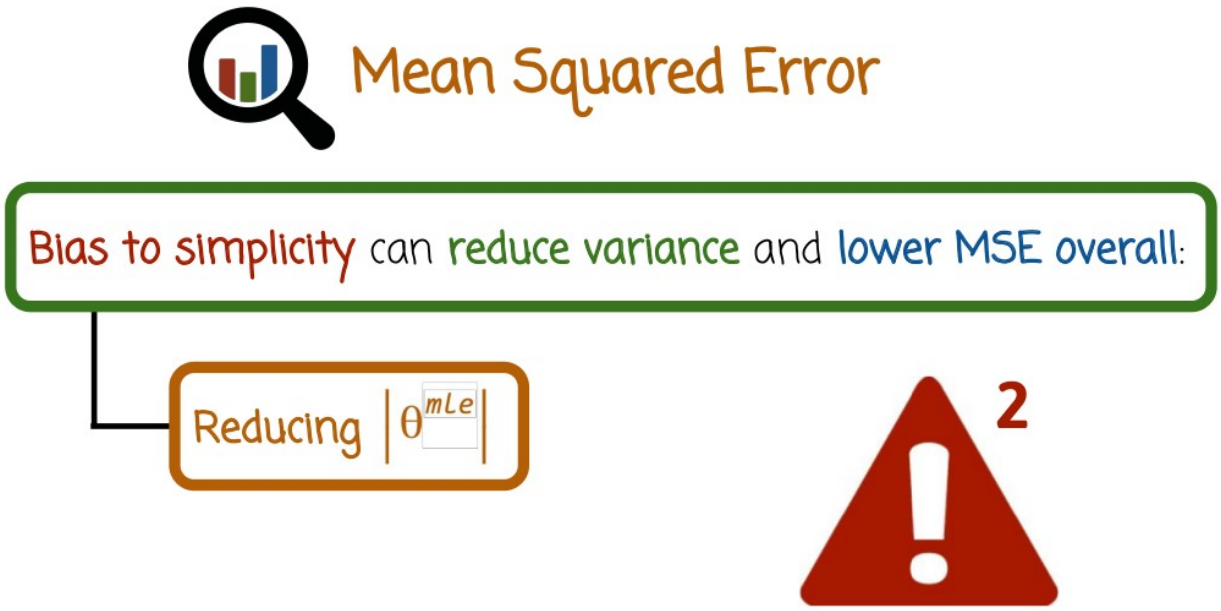


We saw earlier that the mean squared error decomposes of the sum of the bias squared and the variance.

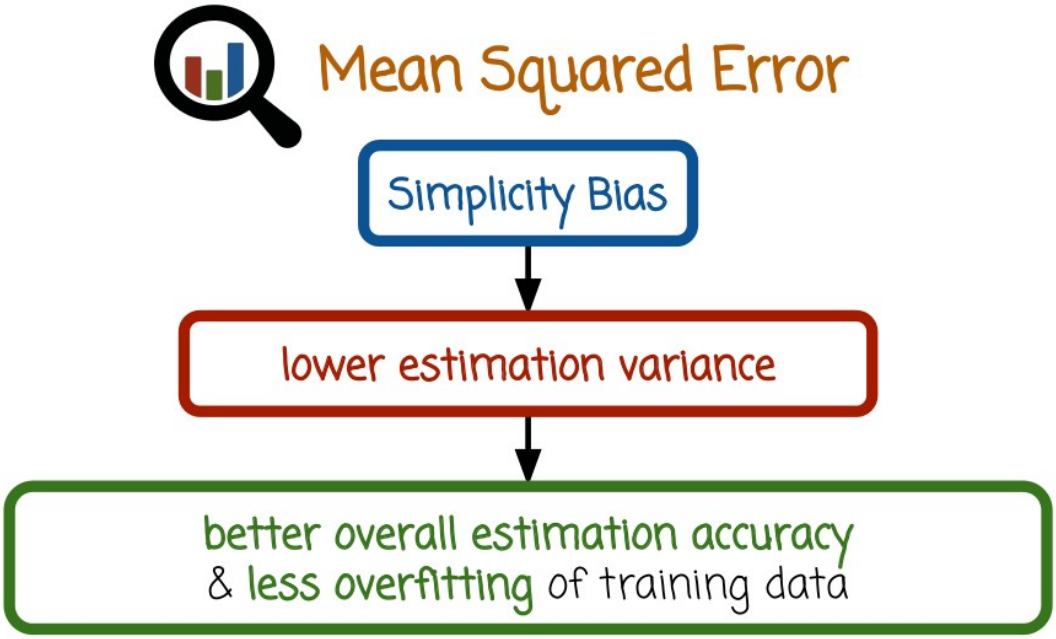
And that the MLE for linear regression is unbiased, which means the bias squared is zero, the only component of MSE will come from the variance. However, we can still have a case of overfitting. And overall, even though bias is zero, variance could be quite high and high enough so that the MSE is high. But if we control for overfitting using regularization, we may be able to reduce variance. Even if we introduce some bias in a way that the sum of the bias squared and variance will be lower than the original case of the MLE. In general, by the way, the MLE may be biased, but still with regularization we can increase the bias even more of the MLE. We can make the MLE more biased, so it looks counterintuitive. Why would we want to increase the bias? In some cases, that's actually a good idea because it will reduce the variance even more. And the sum of the bias squared and the variance will be lower once we introduce that additional bias. Because the decrease in variance will be bigger than the increase in bias that we cause by regularization.



This corresponds to this figure we saw before, where we can look at the case of high model complexity or overfitting. And if we reduce the model complexity, we may increase the bias, but we're going to lower the variance. And what matters is not that we add a new bias, but rather whether the bias that we add is going to be more or less than the variance that we save.

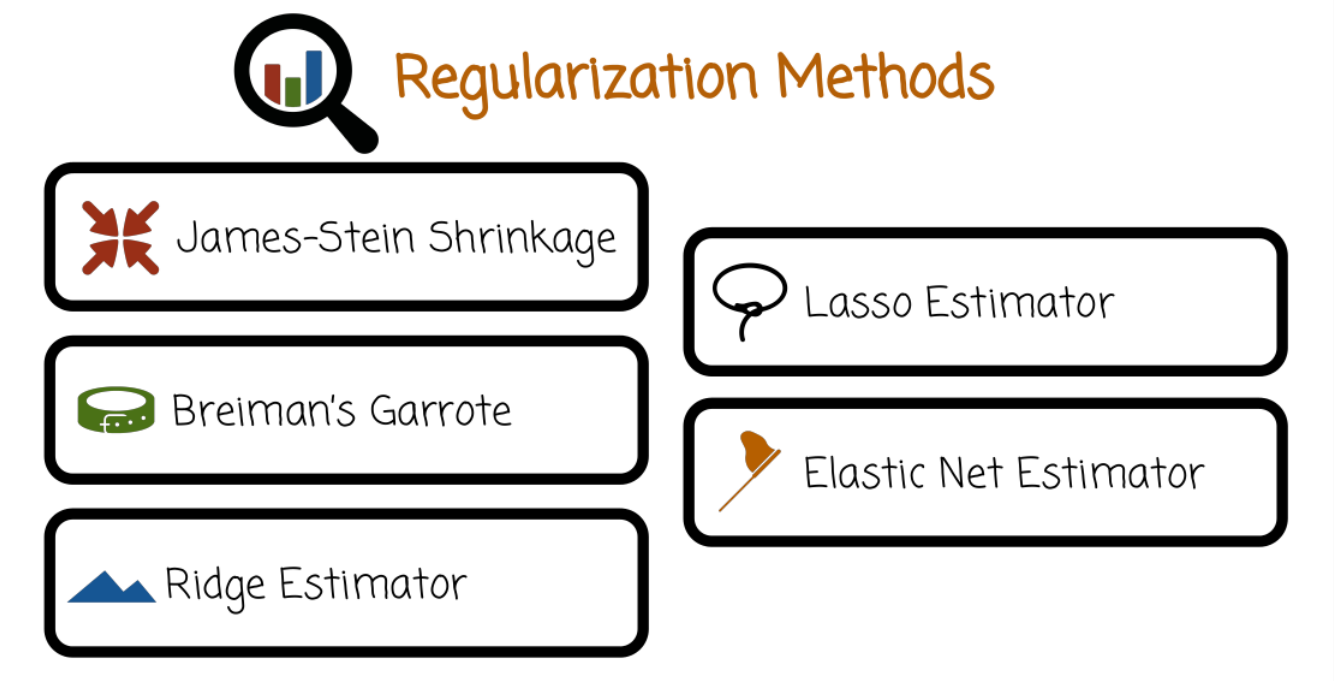


So what does it correspond to in the case of linear models? Bias to simplicity would correspond to reducing the values of the parameters of the estimator, for example, of the MLE. We look at the absolute value of the coefficients that we learned, and we just reduce them, or shrink them towards zero. That will mean that the model places less importance on the corresponding feature. And if we do that in a systematic way, even though the number of parameters, number of features, stay the same, the importance of the features is reduced. And overall, the model will become simpler, less complex, and less likely to overfit the data. And overall in the cases of high-dimensional data, may actually achieve better prediction error in the future.



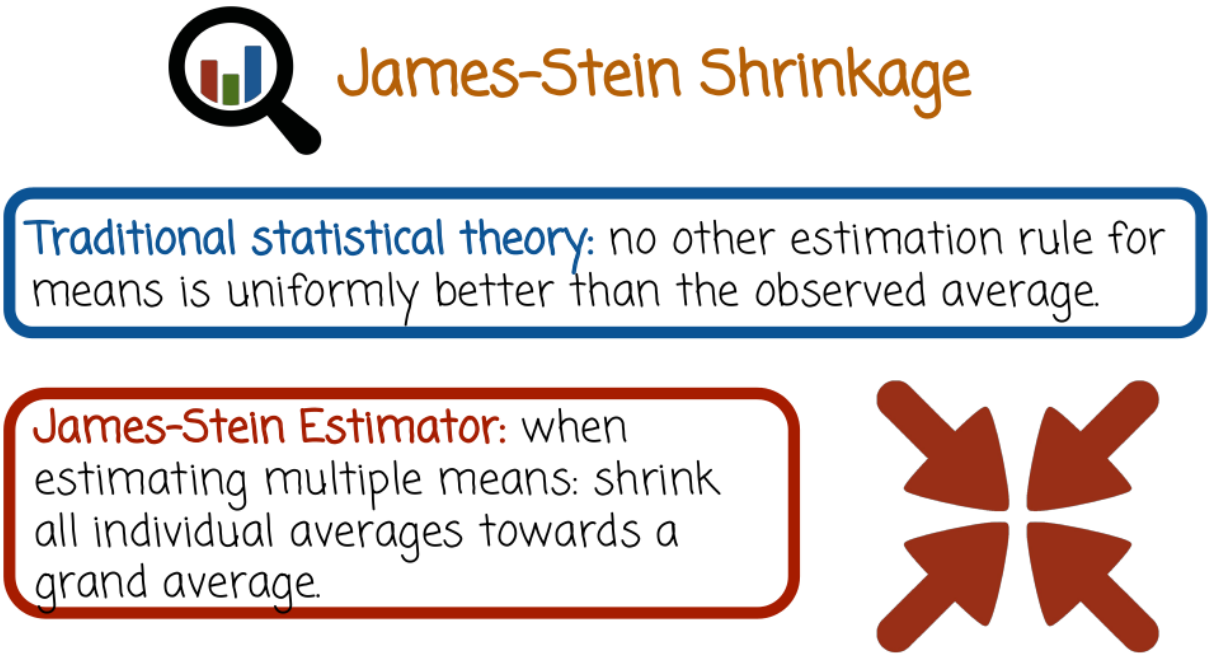
We have simplicity bias, bias to simplicity, which we can think of as reducing the parameters of the model towards zero, reducing their absolute value. That does increase the bias, but it will also lower the variance possibly more than the additional bias, and may result in better overall estimation accuracy and less overfitting.

13 - Regularization Methods

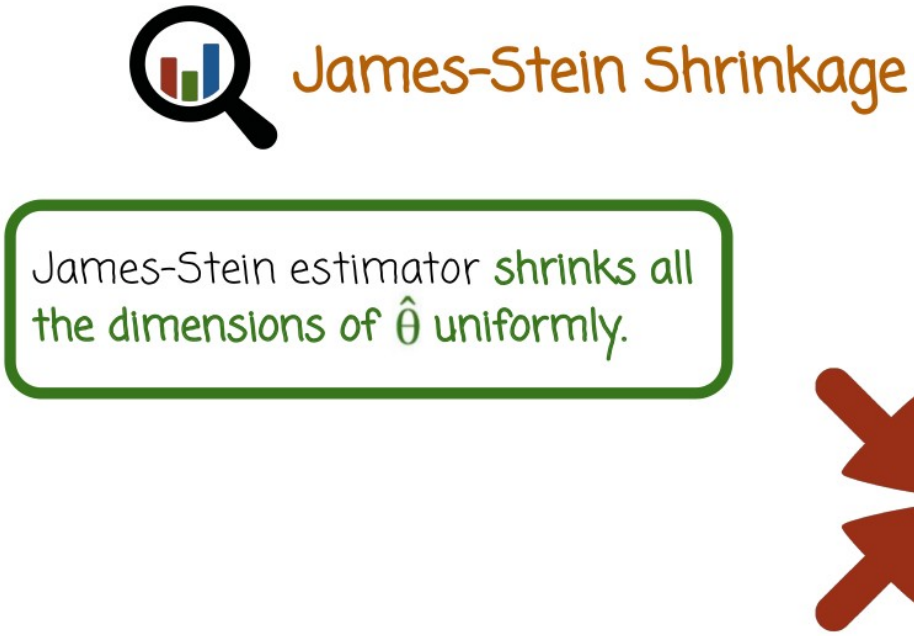


We're going to look at several regularization methods. First we look at the James-Stein Shrinkage. Then Breiman's Garrote. Ridge Estimator. Lasso Estimator. And the Elastic Net Estimator. All of these methods are meant to handle the cases where the traditional case in statistics where d is much lower than n does **not** hold. So we may have overfitting, and the regular MLE may not be the right way to proceed.

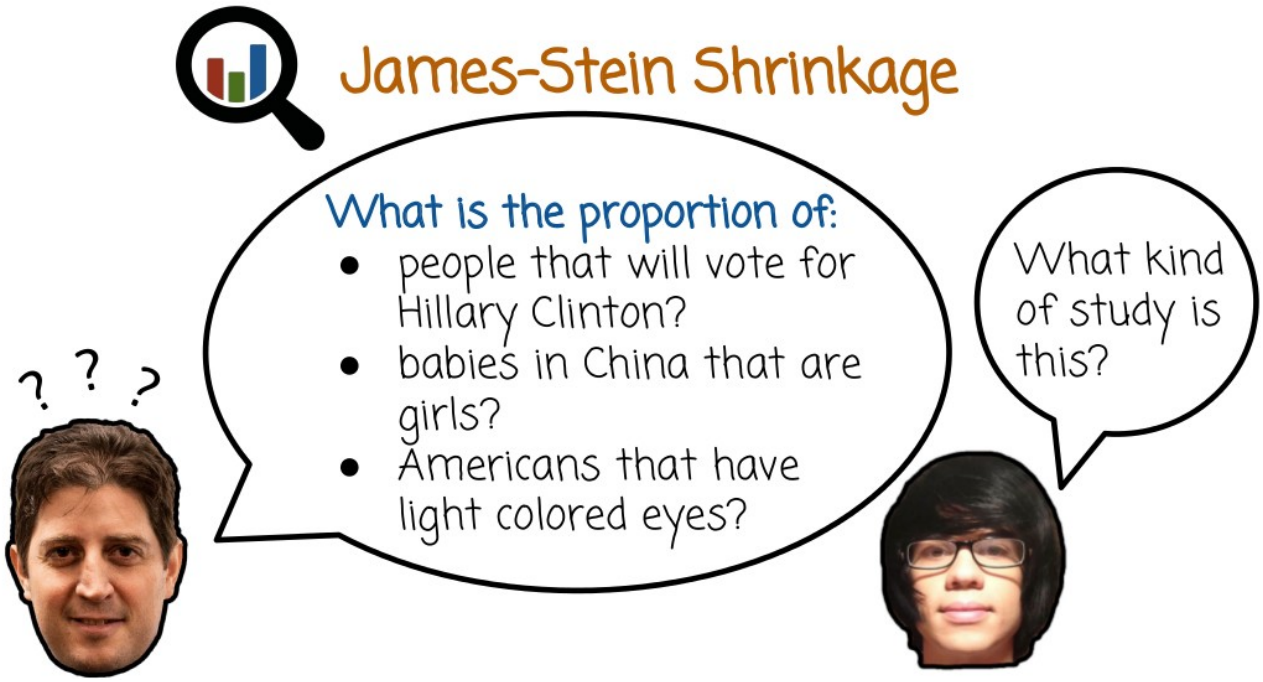
14 - James Stein Shrinkage



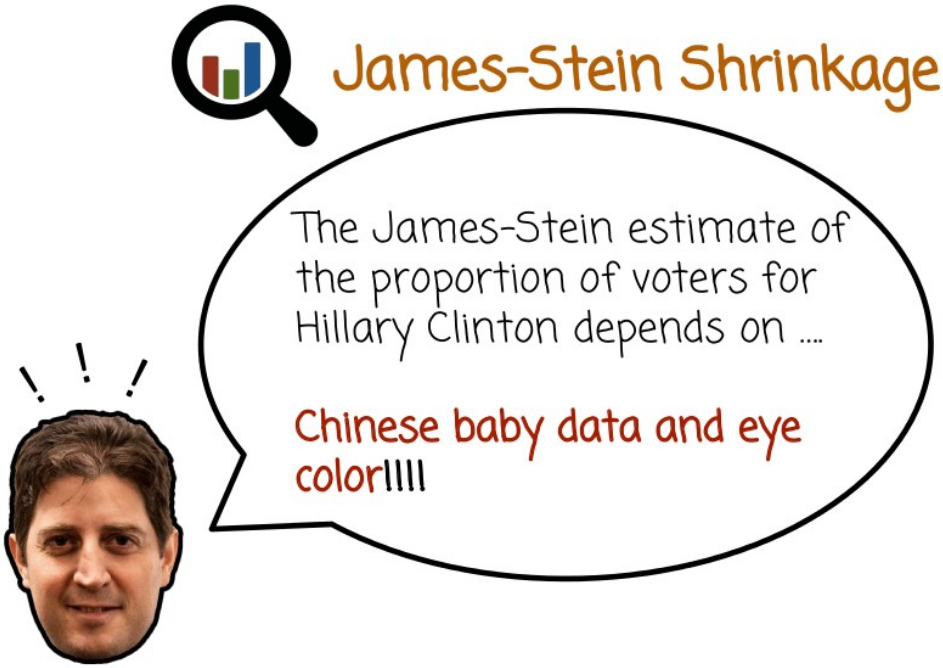
So the first method we'll see is the James-Stein Shrinkage, and we're not going to discuss it, because it's necessarily the right way to do regression or other estimation in high dimension router, because of its traditional importance and the principal behind it, which I think is very interesting, but later on we'll move to more practical methods. James-Stein shrinkage was one of the first ideas or observations that led to the whole field of high-dimensional statistics. In traditional statistics theory, there's no better way to estimate the mean or the expectation than just computing the average of a set of numbers. But the James-Stein Estimator is addressing a situation where we have to estimate multiple means at the same time. So means of different random variables and we want to estimate the means simultaneously. And rather than do the obvious thing which is just compute the mean of every observation and use that to represent the expectation of that variable rather than that, compute the separate means but then also the compute the grand average and shrink all the individual averages toward the grand average. This was very surprising first when it was proposed but there is theory that shows that that's actually performing better than the traditional way of just computing the average separately when you have multiple variables that you're trying to estimate the means for.



One characteristic of James-Stein shrinkage that we'll see is not necessarily the same in the other regularization methods is that it shrinks all the dimensions of the estimators uniformly towards the grand average. And so that could be seen as a limitation of the James-Stein estimator. It does not shrink different averages in different ways, which we'll see later. It's something that we actually want to do, because it's some signals are stronger than others. We want to avoid estimating the noise but we don't want to avoid estimating very strong signals necessarily.

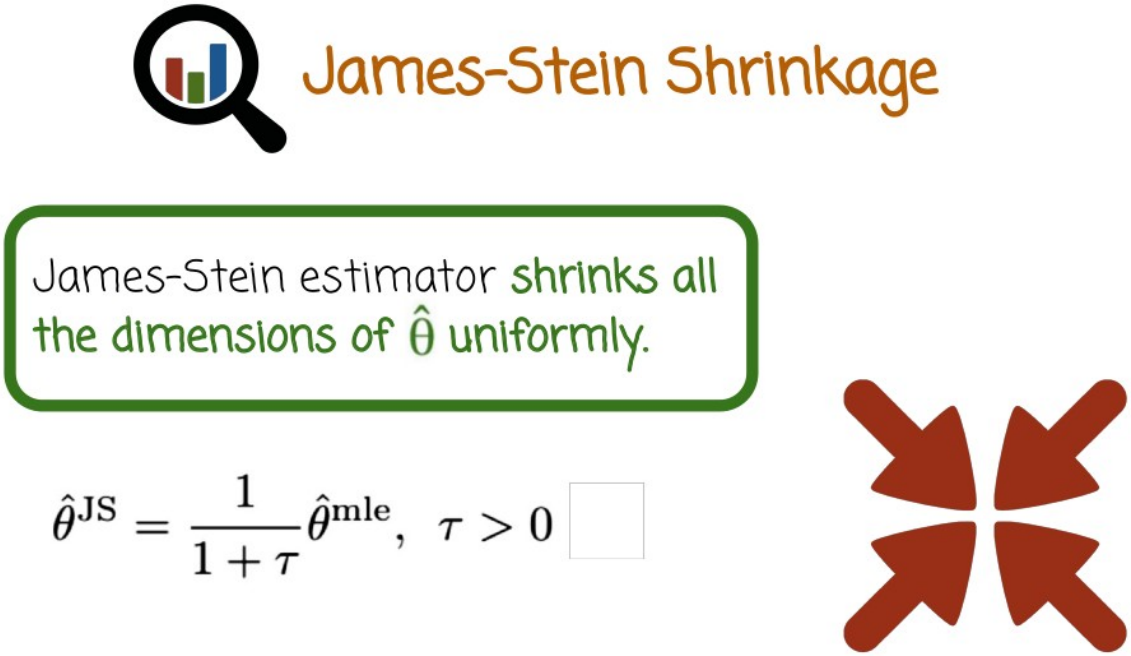


So let's think of an example where we try to estimate the average or proportion of three different quantities. The people that will vote for Hillary Clinton, babies in China that are girls and Americans that have light colored eyes. Serious, why would we want to estimate these three things together. But in the case of high dimensional statistics there some reason to think about that. We going to have a very large number of variables. The variables maybe related to each other very weakly. Maybe not in some cases or not in some cases are. Nevertheless we want to handle such high dimensional data but for the sake of this example let's assume that's what we want to do.



In the James-Stein estimate for example the proportion of voters for Hillary Clinton, because we shrink towards the grand mean, will depend on the Chinese baby and the eye color data sets, which is very surprising.

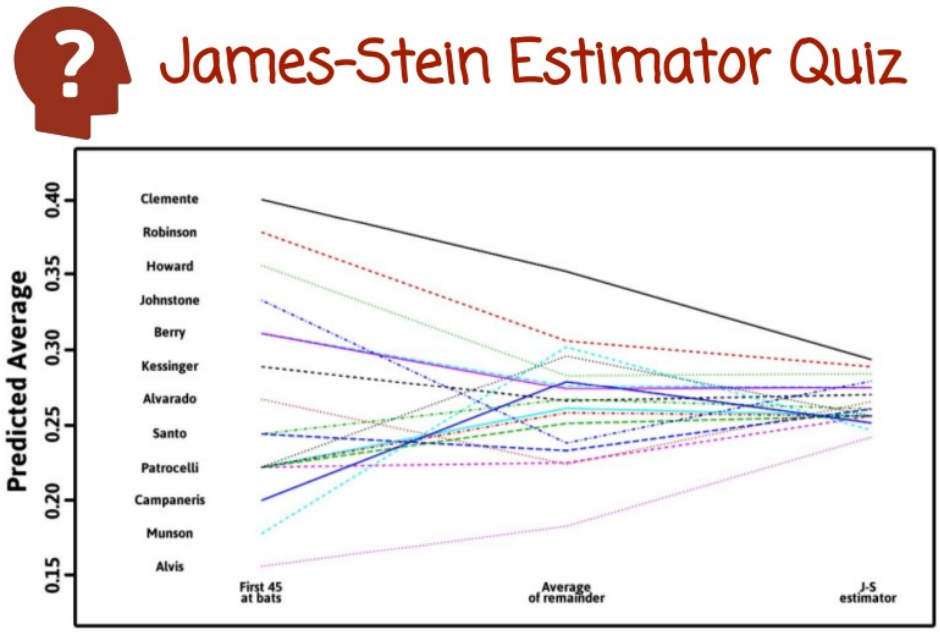
So kind of a big surprise in the statistics world, that this actually makes sense theoretically, and if you're interested in learning more about the James-Stein and its historic effect I'll post some links at the course website are for you to read more about it.



If we want to think of it mathematically more in the context of linear regression or logistic regression, or more generally perhaps, you can think of the James-Stein estimator as an estimator that takes the maximum likelihood estimator and shrink it for example, towards zero. So just reduces it, so multiplying it by 1 over 1 + tau, where tau is a positive number and the this theta hat mle and theta hat JS, JS for James-Stein, these are both vectors. So we have a vector equation here which you can

interpret that as d separate equations, one for each dimension. So each dimension of the mle, we're going to shrink it by multiplying it by 1 over 1 + tau to get the actual value of the James-Stein estimator. So notice that specifically that 1 plus 1 over tau does not depend on the dimensionality of the feature so all features inside this vector are being multiplied by the same coefficient.

15 - James Stein Weakness Quiz



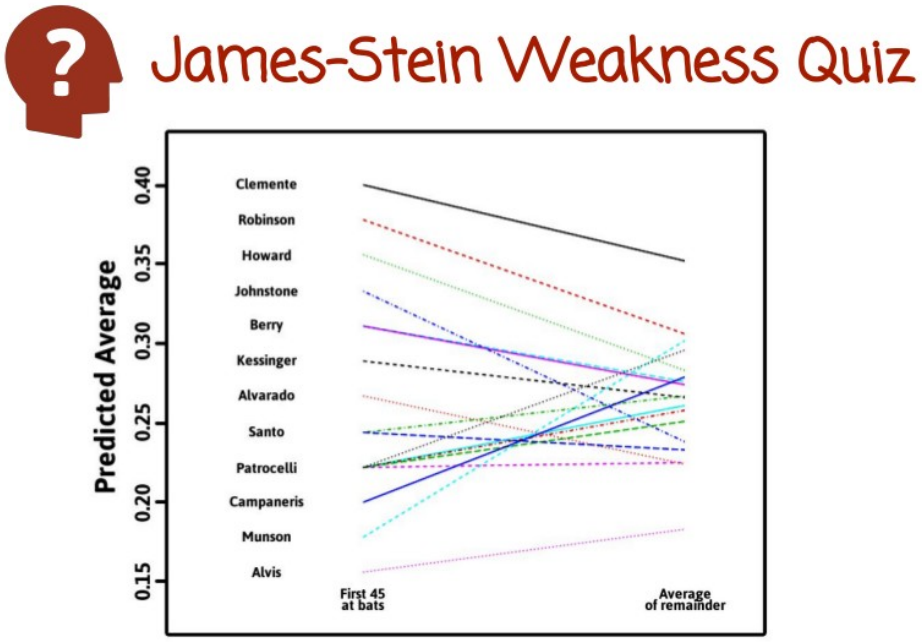
So let's look at a quiz. Let's assume we want to estimate the batting averages of baseball players. And that's actually one of the very famous example that is used to describe the James-Stein estimator. So in this case we have baseball players. And we're going to start the season, the baseball season, and follow the behavior of these players. And we're going to compute their batting average for the first 45 at bats. And that correspond to the numbers here, the left side of the graph. But what we really want is the average batting throughout the season, but we are only at the beginning of the season. So we don't know what is the long term for the season batting average. So one way is to just say, okay, the average batting that we're going to estimate for each of the players is to just compute the average over the first 45 at bats. That's the traditional statistic solution, complete multiple averages. The James-Stein estimator would say, no, let's also compute the grand mean. So the mean of all the means together and then shrink all of these means after 45 days towards the grand mean resulting in the estimator here on the right. The actual truth or what we actually want to estimate that we don't know at the time of estimation is the average of the remainder. We see that it's neither the same as the average over the first 45 at bats, neither the result of the James-Stein estimator, but the question of which estimator is more accurate or is closer to what we want to estimate. And in this case, you see that the average is indeed shrunk towards the grand mean. So the idea of taking the initial averages or means and shrinking them towards the grand mean like the James-Stein estimator is actually the correct thing to do. Although, probably we could maybe use too much shrinking in the James-Stein estimator, that was the right direction to go. And intuitively, you can explain it by there are some extra variability that is just due to the sampling and the fact we don't have enough data and the long term behavior of these baseball players would be closer to the grand mean rather than reflected by the initial performance after the first

45 at bats. This is a famous example and we're going to link in the course website to more information about that.

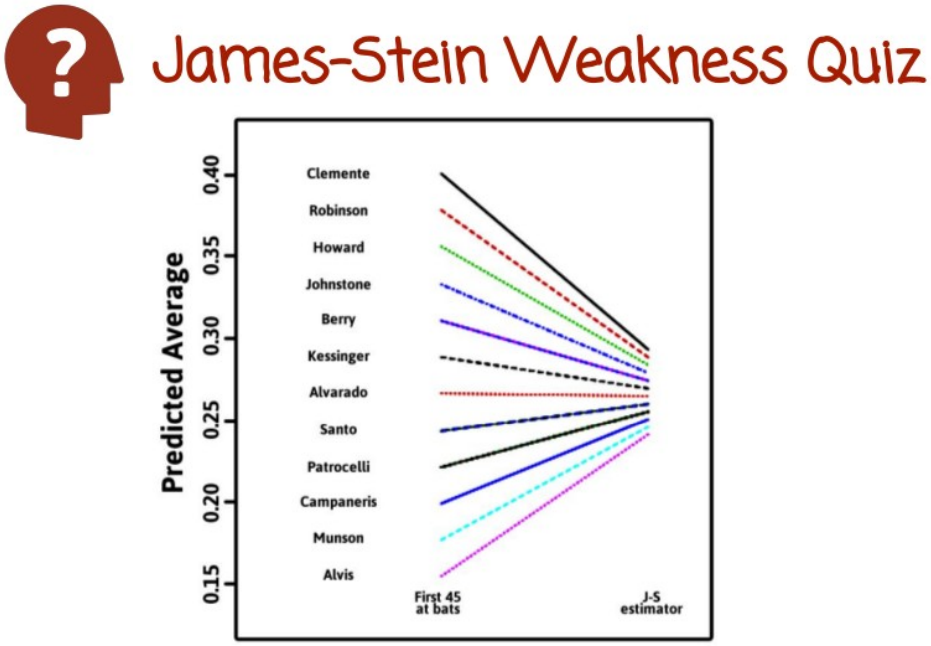
What is a weakness of the James-Stein Estimator?

What is the weakness of the James-Stein Estimator?

16 - James Stein Weakness Quiz Solution



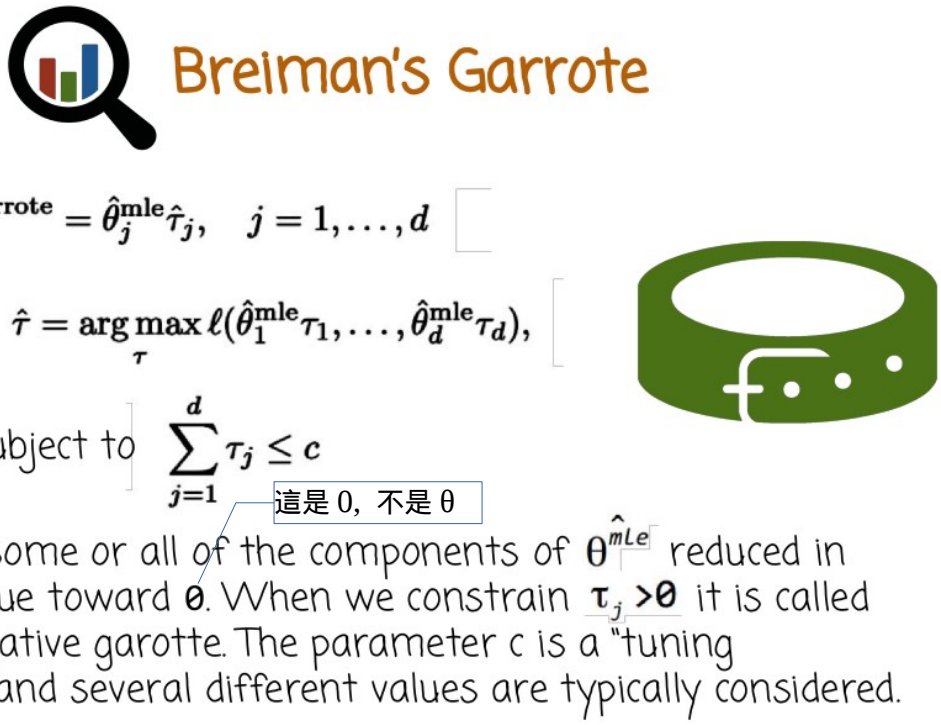
Let's look at the plot of just the first 45 at bat averages, which we see here on the left side and then the average for the remainder of the season.



And then look at what happens with the James-Stein estimator. The relationship between the first 45 at bats and the James-Stein estimator. We see here very clearly the fact that all the individual dimensions or averages are being shrunk in the same amount, one plus one over tau, which is clearly not the real situation.

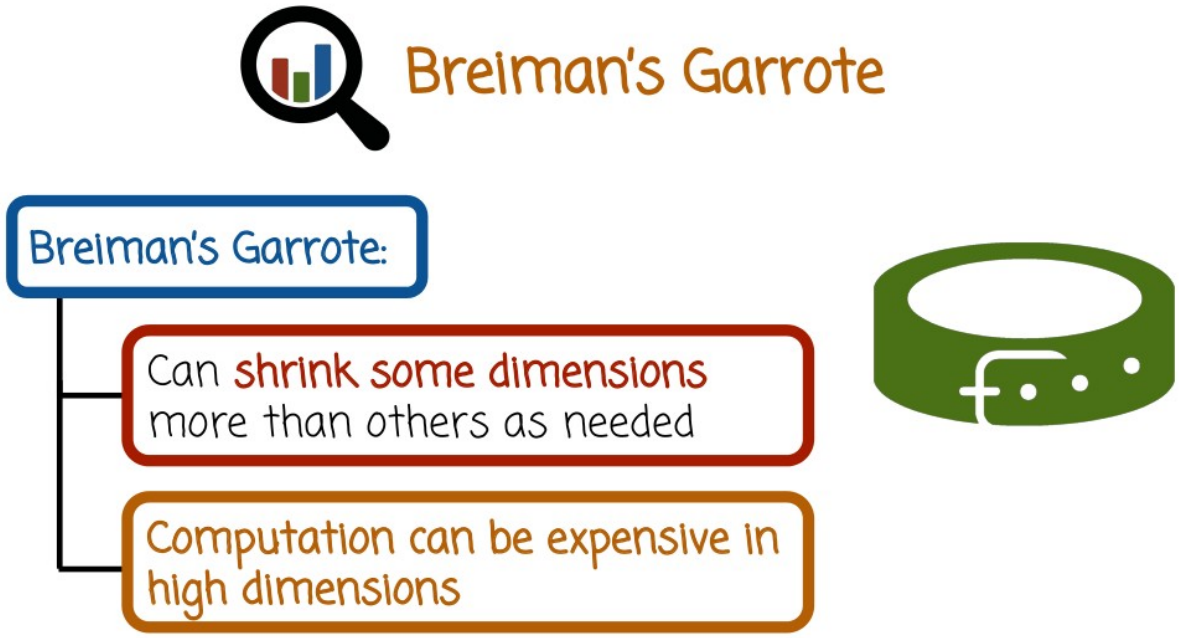
If we go to the slide that shows this relationship between the first 45 at bat averages and the actual average which we're trying to predict. We see that the shrinkage does happen, but it's very non uniform, so correct shrinkage in some cases it's weak, in other cases it's large. Sometimes the ordering between the dimensions or the players get reversed. So in this case, two lines may cross as they map the initial average to the final average. James-Stein estimator cannot capture that.

It will keep shrinking all the players or in the case of parameters of linear regression, logistic regression, will keep shrinking all the dimensions the same. Towards the grand mean or if we want to control over fitting in linear regression towards zero. But it will not allow the data to show us that the shrinkage should be done in different amounts for different dimensions. For example, if some dimension appears to be less relevant maybe to be shrunk more than the others.
17 - Breimans Garrote



Let's move on to the next method Breiman's Garrote. Breiman's Garrote brings the maximum likelihood estimator unlike James Stien shrinkage in a way that is different for each dimension. So, we're going to look at the J dimension of the mle vector and we're going to shrink it by a component or a coefficient constant called tau j. And so as j goes from 1 to d, some tau j may be bigger some are smaller and different dimensions of the mle could be shrunk more or less.

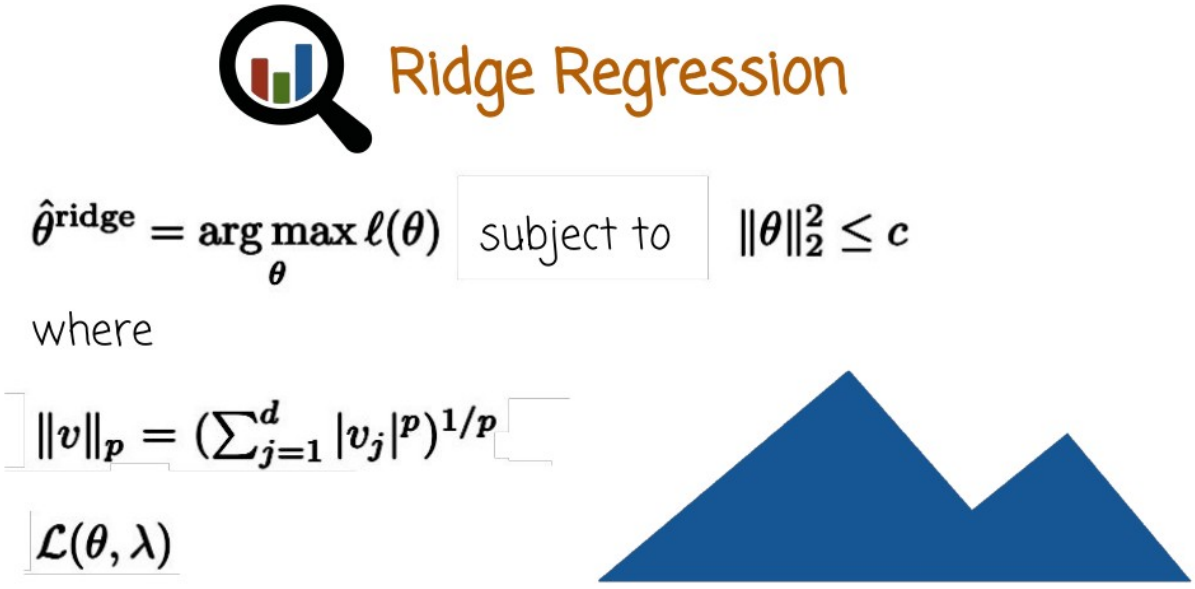
How do we know what is tau? So to find tau, we need to solve a separate optimization problem, which is similar to maximum likelihood. So basically, we find the tau that maximize as the log likelihood where the vector of parameters instead of theta 1 to theta d is theta 1 times tau 1 to theta d times tau d. Subject to a constraint that the sum of the tau js are less than or equal to some number c. When c is less than d, some or all of the components of the mle may be reduced absolute value towards 0. In addition to the constraint right here, we also constrain tau j to be no negative. In this case, this is called the non-negative Garotte. That's the usual setting of the Garotte. The parameter c is a tuning parameter and we may want to try different values, and then test the solutions corresponding to the different values of c. Test them on held out data and then select the model that performed the best on the held out data. This is a common situation in regularization, usually you have one or two hyperparameters or tuning parameters like the c here. And you don't know what's the best selection, so you just try different values. And typically, people form a grid of values and the Garrote, in this case will be solved for every values c the grid of values evaluated on a held outset and we're going to choose the model that profound best on the held outset. One thing again, I want to emphasize is that in this case, the shrinkage is learned from the data. And not only it's learned from the data, different components, different dimensions maybe shrank differently.



So as we discussed earlier, some dimensions may shrink more than others and that's going to depend on what the data shows as much is needed. One issue is however is we need to solve an optimization problem and other optimization problem, the Garrote optimization problem can be expensive in very high dimensions. And so, there is some cost associated with that and maybe more costly than just computing the. So, there's some extra costs. In low dimensions, it's typically not a problem. But remember, the prime in this case for regularization and the garrote, specifically is high dimensions. And so if the dimensions are very high and the computational cost is very high, we may not want to use Breiman Garrote in that case.

後面講的 Ridge Regression 就是 L2 norm, Lasso Regression 就是 L1 norm, Elastic net estimator 就是 L2 + L1 norm. 沒有別的了.

18 - Ridge Regression



Our next method is called ridge regularization. It was originally similar to other garrote and James-Stein. They were all proposed, for James-Stein for means and garrote for regression. Ridge also originally proposed for linear regression but the same principle can apply for other models like logistic regression and also other methods.

So here is the definition of ridge regression. We going to maximize the likelihood or the log likelihood, subject to a constraint. And the constraint is that the l2 norm of the vector theta is less than some constant c. The l2 norm squared right here is just the sum of the squares of the dimensions of theta. Remember, theta is a vector, so basically that expression right here, l2 norm squared, is theta 1 squared plus theta 2 squared plus theta 3 squared, all the way to theta d squared. More generally, you can think of the p norm, in this case it's the l2 norm, but the lp norm, we have the absolute value of each component of the vector raised to the power p, summed up and then taken to the power 1 over p. You can do the math and substitute 2 for p, and you get that what we saw earlier, because it's raised to the second power, it cancels with the square root. You just get the sum of squares. So this optimization problem here is formulated in a way that's not necessarily very easy to solve because it's a constraint optimization problem. And what we're going to do is we're going to reformulate it using something called the Lagrangian, which is the tool from calculus that I hope some of you remember from calculus course. If not, I will include some reference material for that.

l 就是 optimization function

Basically, the Lagrangian is defined as the optimization function minus the constraint. In this case, the l2 norm minus c, is less than, equal to 0. That's the reformulation of the constraint multiplied by the Lagrange multiplier, lambda.

Equivalence can be seen using Lagrange multipliers

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_2^2 \leq c \quad \Rightarrow \quad \hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_2^2, \qquad \lambda \geq 0$$

$$\mathcal{L}(\theta, \lambda) = \ell(\theta) - \lambda(\|\theta\|_2^2 - c)$$

$$\nabla_\theta \mathcal{L} = 0$$

And then what method of Lagrange multiplier is just telling us that if we want to solve the original problem written right here, that's going to be equivalent, instead of solving it, we can just solve a different problem which is here on the right side, which is simpler. And this is just setting the derivative of the Lagrangian to 0 with respect to theta. So the Lagrangian is the log likelihood minus the Lagrange multiplier time the constraint. And the constraint is a little bit different from here, because we moved c to the left-hand side, and so the constraint is just something less than or equal to 0. And that's the Lagrangian right here. And we set the gradient with respect to theta to 0. And we find the stationary point, and when we do that, that will give us the solution to the optimization problem that is also the same solution as the solution to the original problem right here. But the difference is that this is much easier to solve computationally because we can just take gradient and do, for example, stochastic gradient descent or gradient descent. Or we can even just use matrix operation to find a closed form solution as we'll see later. This problem right here has a closed form solution, you don't need some sort of gradient descent, however, in some cases you may want to do that, nevertheless.

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_2^2 \leq c \quad \Rightarrow \quad \hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_2^2, \qquad \lambda \geq 0$$

$$\nabla_\lambda \mathcal{L} = 0$$

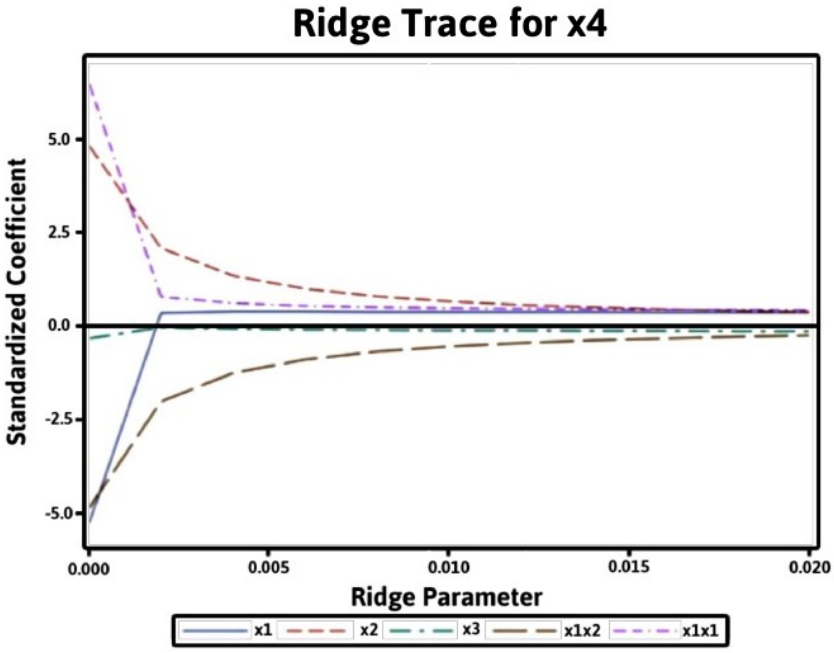
So the middle of Lagrange multipliers is also telling us we need to find the stationary point of the Lagrange with respect to lambda or set the gradient with respect to lambda to 0. But we don't need to do that in practice, because we don't care necessarily about achieving the lambda that will correspond exactly to that c threshold right here. Because we don't know what's the right threshold c. And we don't know what's the right lambda.

Solve for different values of lambda, evaluate each solution on hold-out set and select model that performs best

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda \|\theta\|_2^2, \qquad \lambda \geq 0$$

So, in practice what we do is we basically solve this optimization problem right here, for different values of lambda on a grid of values. Evaluate each solution on a hold-out set and then select the model or the lambda that performed the best on the held-out set. This is not that much more expensive than a regular linear regression or logistic regression. If you are willing to solve a linear regression or logistic regression in high dimensions, adding this penalty right here is almost for free. Whether you use SGD for logistic regression, or if it's linear regression, you can use either SGD or find the closed form solution. It's basically the same cost and the solution will perform often times better than just the MOE, especially when you have high dimensional cases. And in particular, if you are careful at evaluating for different values of lambda on a specific grid that is sufficiently fine spaced grid and picking the best solution on a hold-out set

19 - Ridge Trace



So, let's look at a type of graph that's called parameter trace and what we have here is an x-axis corresponding to the ridge parameter or the lambda. As we increase the lambda, we're going to penalized more and more large values of theta. And as you see as we increase the ridge parameter, the coefficient go down towards zero and that's a form of shrinkage just like James Stein or Garrote. We're going to reduce the coefficient that we learned, the theta i's that we learned by the mle. We're going to reduce them towards zero. And by doing so, we're going to control or reduce the mode of complexity and reduce the amount of over fitting that we do. Looking at this graph, we can see the dependency of the coefficient size on lambda and how much as we go to the right, the coefficients would go down and that's very insightful to see at what pace the model complexity is contained in terms of the ridge parameter.

20 - Estimator Comparison Quiz

Model: **E(Y) = 0 + X1 + X2 + e ; e ~ N(0,1)**

The measured variables are: **x1,x2,x3**

x1 and **x2** are **U(0,1)**
x3 = 10 * X1 + unif(0,1)

corr(X1,X3) = sqrt(100/101) = 0.995

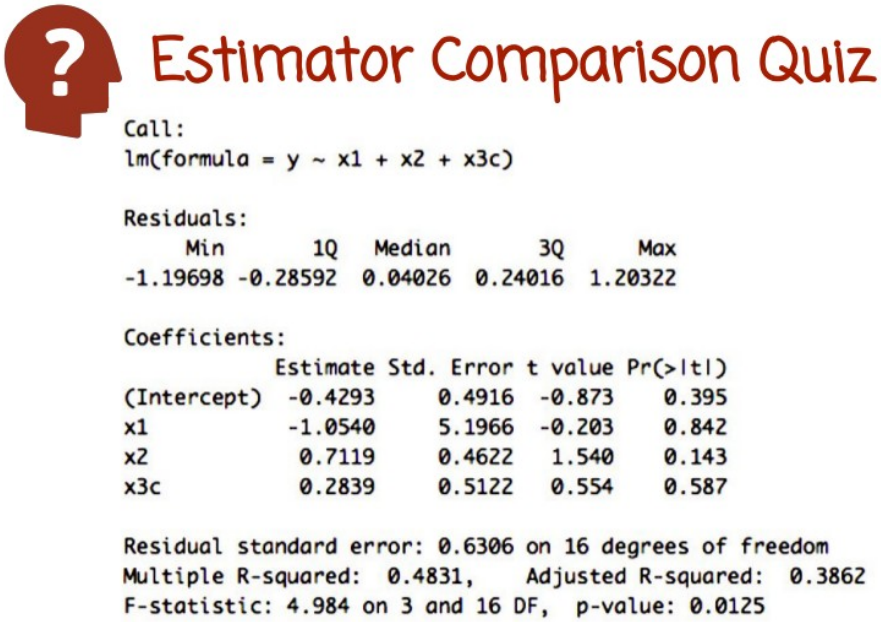
So let's look at an example of ridge regression and linear regression. We'll start with linear regression. So we have here model where we have a label Y generated by X1 + X2 + Gaussian noise. X1 and x2 are two variables that are uniform between 0 and 1. And so we take two uniform variables between 0 and 1 and we're going to add to them Gaussian noise. That's going to give us the variable Y. Then we're going to create another variable, x3, which is 10 times X1 + uniform random variable between 0 and 1. And notice specifically the correlation between X1 and X3 is very high. And what we're going to do in the quiz is estimate linear regression model based on X1, X2 and X3. And examined the model summary that you get from the linear regression model in R.

21 - Estimator Comparison Quiz Solution

OLS fit of 3-variable model using correlated x3.

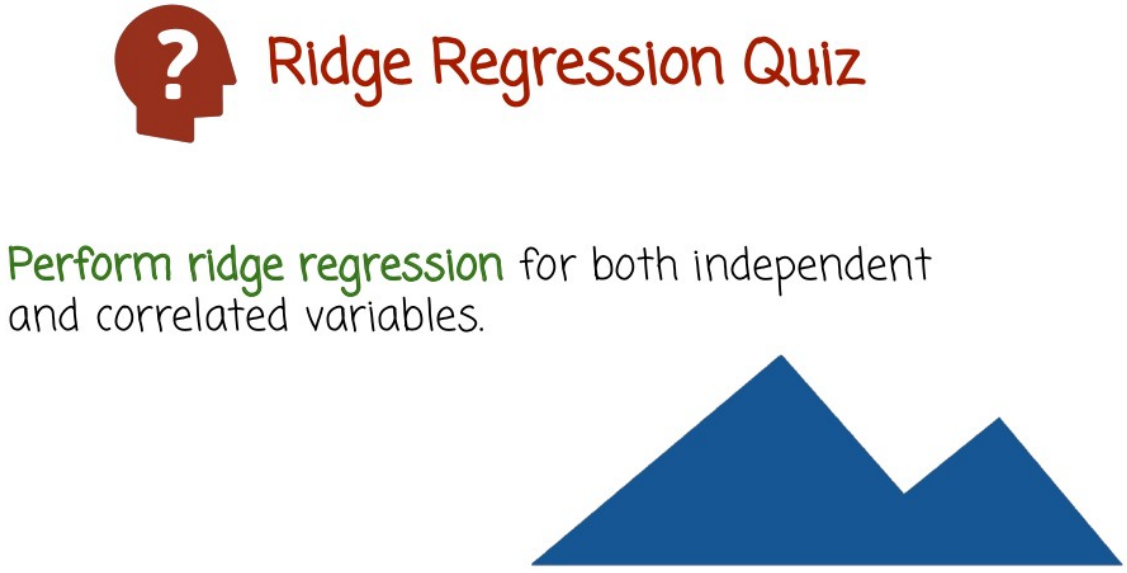
```
olsc <- lm(y~ x1 + x2 + x3c)
summary(olsc)
```

So here is the commands that I use. You can probably do a little bit differently using the lm function which builds a linear regression model and we pass it the formula 1 to the x1 + x2 + x3c. And then we look at the summary of the returned object.



In this case, that's the solution. You can look at the residuals and coefficients, and so on.

22 - Ridge Regression Quiz



So in this quiz we're going to use the data from the previous quiz. But instead of building a linear regression, let's build a ridge regression for the same variables, X-1, X-2, and X-3. Please write ridge regression command in R, and then examine the model of summary.

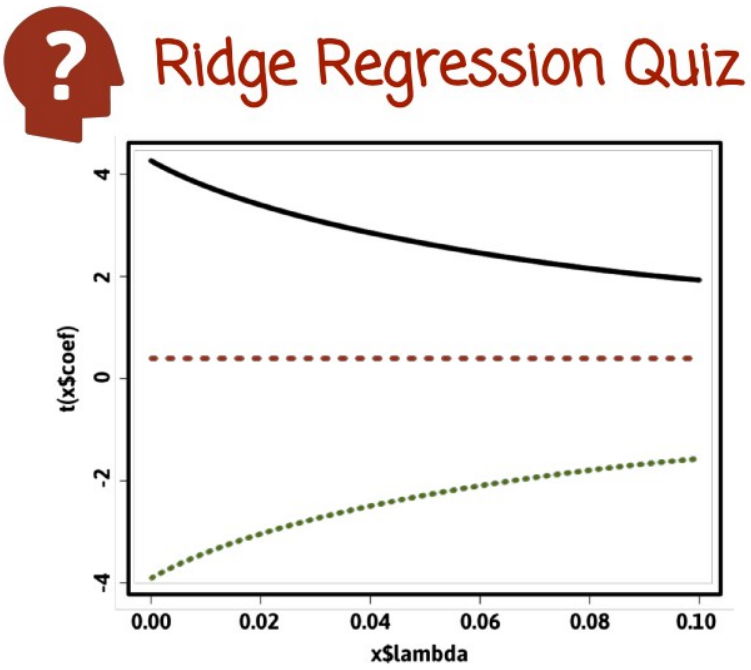
23 - Ridge Regression Quiz Solution


```
ridgec <- lm.ridge (y ~ x1+x2+x3c, lambda =
seq(0, .1, .001))
```

So here is the ridge estimation command, lm.ridge. And we pass it both in the model formula, as well as a grid on which to estimate with different values of lambdas. In this case, lambda is going to be a value on a grid between 0 and 0.1 in increments, so 0.0.1.


```
> summary(ridge)
        Length Class  Mode
coef    303    -none- numeric
scales    3    -none- numeric
Inter     1    -none- numeric
lambda  101    -none- numeric
ym        1    -none- numeric
xm        3    -none- numeric
GCV     101    -none- numeric
kHKB      1    -none- numeric
kLW       1    -none- numeric
```

And we can just type summary to see the summary of the object return by the ridge estimation function.



If we want to plot the ridge coefficient as the function of lambda we can do so. And if the x-axis is the lambda and the y-axis is the coefficients. We see here there are three coefficients and they generally are shrunk toward zero in different ways. The red line is almost stationary, but the other two coefficients are reduced towards zero as we increase the lambda coefficient.

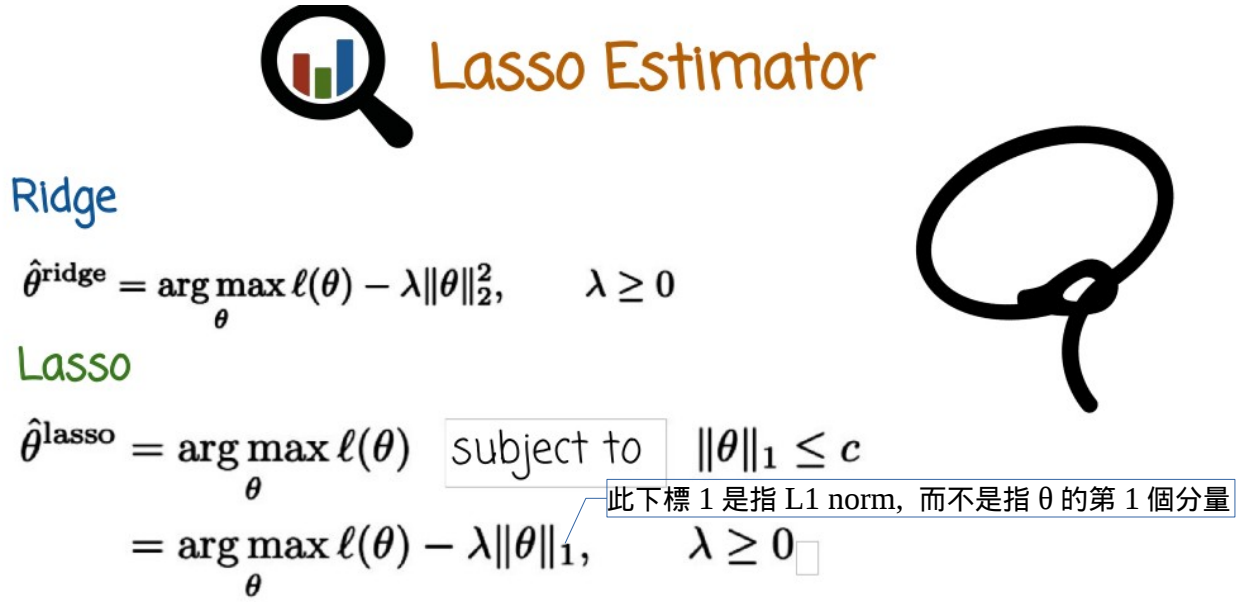
24 - Lasso Estimator

LASSO = least absolute shrinkage and selection operator

Model	Characteristic
Ridge	Must include all or none of the coefficients
Lasso	Does parameter estimation **and** variable selection

We'll see now the LASSO estimator. So it's similar to ridge in the sense that it's a penalized maximum likelihood solution, but the penalty instead of being L2 is going to be L1. From an intuitive prospective, ridge estimator will include all of the coefficients or all of the parameters of the maximum likelihood estimator in the ridge estimator of the same parameters will be present, although smaller or closely to 0.

In the case of LASSO, some parameters will be shrunk to zero precisely. And by doing so, we basically perform variable selection, because if a parameter is mapped or sent to 0, the corresponding feature is no longer used in the model, so it's basically removing some features from the model that will no longer be used. Again, with the ridge regression, we're not doing variable selection, we're keeping all the original features. The reason there is a none caveat here is because it may send all values to 0. If the penalty lambda is just extremely high, infinity, all the values will be 0. But more realistically, it's not going to map any parameter to 0 that is not normally 0 by DMLE. So ridge regression reduces the values of the theta i's that we get from a normal maximum likelihood solution. While LASSO both reduces the magnitude of the theta i's from the maximum likelihood as well as sends some of them to 0 or removes some of them altogether from the estimation model.



So let's say mathematically, this is the ridge, that's the dual formulation or the Lagrangian formulation. If you remember, lambda is the Lagrange multiplier. And so we solved the ridge regression by maximizing this quantity here. And we have penalty term that penalizes large values of theta j's, theta i's, and the components of theta.

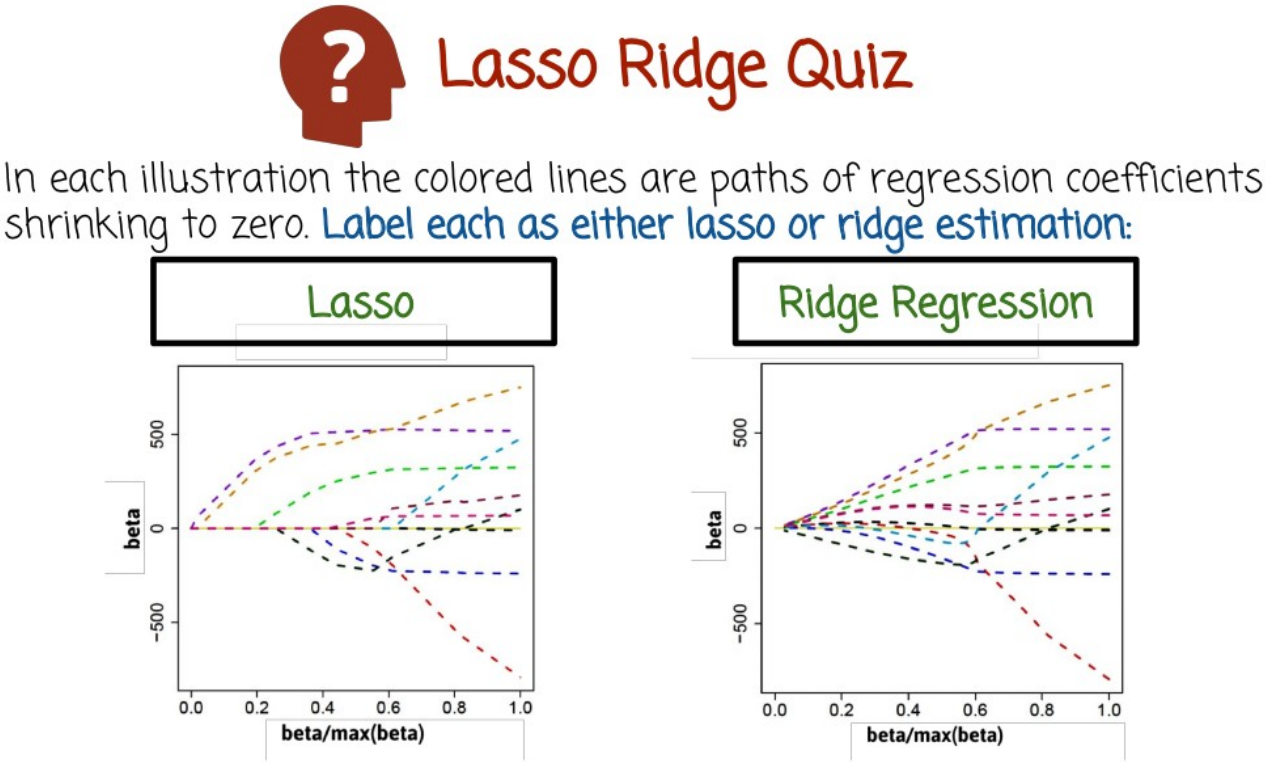
LASSO is very similar, but instead of having a constraint of l2 norm, we have a constraint of l1 norm. And now to parse this right here, go back couple of videos to the definition of an lp norm, and you will see that this l1 norm right here, what this corresponds to is just the sum of the absolute values of the components of theta. This is the primal formulation or the constraint optimization. It's not easy to optimize directly, so what we're going to do again is we're going to use Lagrange multipliers to go to the duals formulation right here. And we're going to solve the dual problem right here, maximizing the Lagrangian minus some sort of a penalty. The penalty this time will be l1 penalty. So the sum of absolute values instead of the sum of squares. So it's a different penalty, does not look like a big difference, it's almost exactly the same as ridge. We maximize the likelihood, but add a penalty of the sum of absolute values instead of the sum of squares. So it seems like a pretty small change to go from ridge to lasso.

Penalty	Model	Result
l_1	lasso	Encourages sparse solution. For a given some of the components of lasso will be zero
l_2	ridge	Does not encourage sparsity. The coefficients of ridge may be close to zero but not precisely zero.

But as we saw before, there is actually fundamental difference between l1 and l2. l1, which is used in lasso, encourages sparse solutions. Sparse meaning many components will be set to zero. Whereas the l2 does not encourage sparsity. Although in both cases, the magnitude of the theta j's will go down, in l2 or ridge model, they will be nonzero, whereas in l1, they will go down to actual 0, some of the components, at least. And knocking them to zero has the advantage of making the model simpler and also computationally faster to evaluate later on, when you try to predict new dataset. If you have very high dimensionality to begin with, let's say you have a billion dimensions to begin with and you do l1 lasso. Maybe 90% of the billion dimensions will be mapped to 0. And then when you try to predict very, very quickly the value of a new label, instead of adding up a billion numbers, you'll just have to add 10% of a billion numbers, which would be much faster. So the sparsity has significant advantages in high dimension in terms of being computationally easier to evaluate as well as the model becoming more interpretable.

And notice in addition for encouraging sparsity in l1, we also see a reduction in the values of theta, theta i, similar to ridge. So it's not just selecting some variables and for the other variables just do the MLE. Rather, we select some variables that will be zeroed out, and for the other variables or the other parameters, they will still be shrunk towards 0.

25 - Lasso Ridge Quiz



In each illustration, the colored lines are paths of regression coefficients as a function of the penalty constant. The regression coefficients are being shrunk to zero. One of these graphs is the result of lasso estimator. The other one, ridge estimator. Label one is the lasso, and the other one is ridge.

26 - Lasso Ridge Quiz Solution

The first case we have a lasso solution. The second case we have a ridge solution. How do we know that? As we change the value of the regularization parameter, we see that more and more of the parameters will actually map to zero or go down to zero accomplishing sparsity. Whereas in the case of ridge, changing the value of beta would basically reduce the parameters closer and closer to zero, all the theta I's, the components of the vector theta, but it will not zero them out. You see here in the case of Lasso, many of them will actually be mapped or sent to zero when you see the curve actually converging or intersecting the x-axis.

27 - Linear Rigression Part 1

Linear Regression

The methods in this lesson apply generally to maximum likelihood problems.

The linear regression setting enables derivation of closed forms which provide insight into the differences between different regularizers.

However, ridge and lasso can also be applied for other models like logistic regression (and they are very successfully applied).

The methods we saw generally apply to maximum likelihood problem. In many cases, they're applied to linear regression and in the case of linear regression we have in particular simpler theory. So we can reason theoretically about the solution of region of so, and so on, but that theory could be much harder, or maybe does not work for more complex models like logistic regression where there is no longer a close solution. The solution is harder to get and the theory is much harder to apply. But in practice, we use both ridge and lasso more broadly for both linear regression, for logistic regression, and also for other methods like deep learning or whatever it's pretty general way of controlling from model complexity.

Linear Regression

Assume:

X: a matrix of training data $X^{(1)}, \ldots, X^{(n)}$

Y : vector whose entries are training labels $Y^{(1)}, \ldots, Y^{(n)}$

Then the conditional log likelihood is:

$$-(Y - X\theta)^T(Y - X\theta) + c$$

Setting its gradient to zero:

$$-X^TX\theta + X^TY = 0$$

Implies:

$$\hat{\theta}^{mle} = (X^TX)^{-1}X^TY$$

So let's see how math simplifies of ridge and the lasso, the math simplifies in the case of linear regression and the theory and the math will give us some insight about what's happening. That, in the case of linear regression, we will not be able to get the same kind of nice results in math for logistic regression or more complex models, but the same intuition applies. So in the case of linear regression to remind you we have a matrix x whose rows are the training observations. The vector is x superscript 1 to x superscript n. Then we have a vector y populated with the training labels. The log likelihood conditional log likelihood is the squared error between the predicted values of y and the actual values. This is represented here in matrix form that's the row vector times the column vector that's a scalar and that's basically the matrix notation for the conditional log likelihood.

If we want to maximize the log likelihood we can set its gradient to zero. Setting it to zero basically means taking the derivative and setting it to zero but little bit tricky to do that in the matrix notation. The right way to see why this equals setting the gradient to zero here and getting this solution is basically to do it individually for each component. Let's see how we can reason about that. So we can multiply y transpose by y. And then x theta, take the transpose of that which is theta transpose x transpose multiplied by y. Take the y transpose, multiply it by x theta and take x theta to get it transposed which is theta transposed x transposed and multiplied by x theta. Then you get something a little bit simpler and then you compute it's gradient, you set it to zero, you get this expression right here. You move this expression to the other side to right hand side, then you multiply by x transpose x from the left on both sides, inverse, x transpose x inverse. And this way you get to isolate theta by itself because the inverse of x transpose x * x transpose x is the identity. So, this thing right here goes away. And what you get is x transpose x inverse times x transpose y is the mle. So we've seen this before without derivation here, there's more information on how to get derivation. This is the close form solution of linear regression.

Linear Regression

Special case of orthonormal training examples:

$$X^T X = I : \hat{\theta}_j^{mle} = X^T Y$$

(the orthonormal projection of the columns of X on Y).

In many cases, insightful to simply find the product by thinking of the training samples as being worth the normal. So, when you take the products of different training samples you adjust zero unless it's the same example, in which case it's one. So that's denoted or formulated in a compact way by writing x transpose x, this is two matrixes multiplied by each other equals the identity. That's not usually what happens. In fact, probably would never happen out of like when you record data, you will not get that. But this does help simplify the problem and see what happens. In this case, the equation simplify and it just sheds light and you can do both theory and get intuition. In that case, that you'll just be able get otherwise. And the intuition and the theory even though they apply strictly speaking only to this case, they can still be insightful more broadly. So if that's the case you can go back to the previous slide and

the mle solution simplifies to the equation right here which is basically the orthonormal projections of x onto y. So this is the solution, close form solution of the mle and the restricted, or the extra constrained case mentioned right here.

28 - Linear Rigression Part 2

In the case of **ridge regularization**: setting the penalized log likelihood gradient to zero

$$-\mathbf{X}^\top\mathbf{X}\theta+\mathbf{X}^\top\mathbf{Y}-\lambda\theta=0\Rightarrow\hat{\theta}^{\text{ridge}}=(\mathbf{X}^\top\mathbf{X}+\lambda I)^{-1}\mathbf{X}^\top\mathbf{Y}$$

So what happens with ridge? Well, you can repeat the same derivation that we did before and said the penalized log likelihood gradients. So that's the log likelihood plus the L2 norm to zero. So this is what we had before, the log likelihood in gradient set to zero. But in this case we also have the gradient of the gradient of the l2 norm. The gradient of the l2 norm is just lambda times theta. Lambda being the regularization parameter.

We set it to 0 and we solve this matrix equation. And we get something very similar to what we had before. The only difference is that this matrix, x transpose x, we add lambda times the identity to it, which the reason that's insightful is that it helps a lot in cases where the x transpose, x matrix, is not invertible. If it's not invertible, you actually cannot compute its inverse. But when you add a diagonal matrix times some value, even if it's a small value, you turn the non-invertible matrix into invertible and the problem becomes much more stable. And that's actually a different way to think about high dimensional linear regression in an algebraic sense, rather than in a statistical sense of overfitting. In an algebraic sense, what happen is that the matrix X transpose X becomes non-invertible, and we regularize or we simplify the model or we reduce its complexity by adding some small number to the diagonal in order to make it invertible and more stable.

Linear Regression

In the orthogonal case:

$$\hat{\theta}^{\text{ridge}} = \frac{1}{1+\lambda}\hat{\theta}^{\text{mle}}$$

Each component is shrunk by the same constant factor

$$\hat{\theta}^{\text{ridge}} = \hat{\theta}^{\text{JS}}$$

And so if you remember the orthogonal case where the samples are orthogonal to each other, well, we can get an even simpler expression of ridge in this case. If you follow the derivation we had before, but you add ridge to it, you get that the closed form solution of the ridge in the case where the data samples are orthogonal is d mle times 1 + 1 over lambda. So basically we take each component of the mle and we shrink it towards 0 by the same constant factor. So this is the same as the James-Stein Estimator if you remember. So in this case, ridge regression is the same as James-Stein. But remember this is just your orthogonal case, number one, which is never the case in practice. So James-Stein explanation or equal shrinkage for each component of the mle is not true in general. So number one is the orthogonal case constraint which is not general correct and number two ridge regression can also apply to other models not just linear regression. It can apply to logistic regression or other more complicated model. And then again you do not have that equivalence that we just derived. You don't have a closed form anymore so you cannot follow the derivation for more complex models.

Linear Regression

In the case of lasso regression:

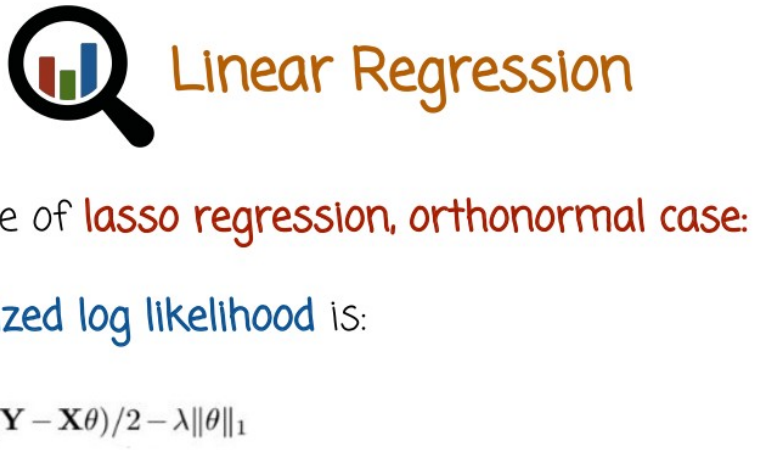
$$0 = \nabla(\ell(\theta) - \lambda\|\theta\|_1) \Rightarrow -\mathbf{X}^\top\mathbf{X}\theta + \mathbf{X}^\top\mathbf{Y} - \lambda s(\theta) = 0$$

where $s_j(\theta) = \text{sign}(\theta_j)$

Let's see what happens with lasso regression. So in this case, things are more complex. We can still get the closed form solution in some cases. In the orthogonal case, but it's going to take us more work and I'm not going to follow every single step of the derivation. I will include reference if you want to follow up the steps of the closed form derivation in the orthogonal case. But let's see what happens. So we take the log likelihood penalized by the L1 norm. That's the objective function that we want to maximize. In the case of lasso, we compute its gradient, then we set it to zero. This is how we solve it. And this is the equation that we get, where s here is the sign of theta. So the thing is that the l1 norm is the sum of the absolute values. And the absolute value, the derivative of the absolute value is just the sine of the number, except at 0 where the absolute value is not differentiable. So assuming we're not at 0 or we do not look at a case where theta is exactly 0, the derivative here is the sign of theta. So this is a vector notation of a vector of plus 1 or minus 1 depending on the sine of each coefficient.

So what we have here is a vector where each component, the j component is the sign of theta j. This is a general case, well, we want to simplify it and get some intuition. And so we're going to move to the orthonormal case.

29 - Linear Rigression Part 3



Endorse a normal case, the samples are orthonormal to each other.

Linear Regression

In the case of lasso regression, orthonormal case:

The penalized log likelihood is:

$$-(\mathbf{Y}-\mathbf{X}\theta)^\top(\mathbf{Y}-\mathbf{X}\theta)/2-\lambda\|\theta\|_1 = (\mathbf{Y}^\top\mathbf{X}\theta+\theta^\top\mathbf{X}^\top\mathbf{Y}-\|\theta\|_2^2)/2-\lambda\|\theta\|_1 = \theta^\top\hat{\theta}^{\text{mle}}-\|\theta\|_2^2/2-\lambda\|\theta\|_1$$

Which decomposes into a sum of d maximization problems:

$$\theta_j\hat{\theta}_j^{\text{mle}}-\theta_j^2/2-\lambda|\theta|_j$$

That can be solved independently

For each j, if $|\hat{\theta}_j^{\text{mle}}| < \lambda_j$ the objective function will be negative unless $\theta_j = 0$ then the objective function = 0

And let's see what happens to the math in this case. So if we expand this expression right here, we get four terms and after some manipulation we get only three terms right here. And the simplification that we get is because of the fact that X transpose X is the identity. And now we further remember the fact that X transpose Y is the mle solution for the orthonormal case. So we're just going to replace that by theta hat mle and what we're going to do next is notice that solving this vector equation. So setting this to zero, we see that it's just decomposes to d separate maximization problems. So we don't have to solve a d dimensional maximization problem, we just solve d separate maximization problems, each one of them one-dimensional. And now notice, this is again emphasizing, this is only in the orthonormal case, which is not normally the case. So we cannot decompose the last optimization into the d maximization problems in general. But we just do that in the case of orthonormal data just to get some more intuition and insight into what's going on. So we get the d maximization promise which can be solve independently. And I'm going to skip a little bit how we solve each one of these maximization problems. Because we do have here absolute value and we're not able to differentiate and set to 0. Because of the fact that the absolute value function is not differentiable. So it's a little bit tricky, we have to pay attention to the fact, is it positive? Is it negative? Is it 0?

Linear Regression

In the case of lasso regression:

If $|\hat{\theta}_j^{\text{mle}}| < \lambda_j$ setting $\hat{\theta}_j^{\text{lasso}} = 0$

Results in a 0 objective function

Linear Regression

In the case of **lasso regression**:

If $|\hat{\theta}_j^{\text{mle}}| > \lambda_j$, $\hat{\theta}_j^{\text{lasso}} \neq 0$

Then:
The objective function is **differentiable**

Set the derivative to 0
$\mathbf{0}$: $0 = \hat{\theta}_j^{\text{mle}} - \theta_j - \lambda \text{sign}(\theta_j)$
To obtain:
$\hat{\theta}_j^{\text{lasso}} = \text{sign}(\hat{\theta}_j^{\text{mle}})(|\hat{\theta}_j^{\text{mle}}| - \lambda)$

Linear Regression

In the case of **lasso regression**:
Combining the two cases we get:

$\hat{\theta}_j^{\text{lasso}} = \text{sign}(\hat{\theta}_j^{\text{mle}})(|\hat{\theta}_j^{\text{mle}}| - \lambda)_+$, where $A_+ = \max(A, 0)$

Therefore:
The sparsity encouraging threshold nature of lasso as it zeros out small coefficients

Recall: ridge will make them smaller, but non-zero.

So let's see what happens when we solve this d separate optimization problems. We get that the diff component of the lasso parameter vector is the sine of the j component of the lasso vector. Is the sine of the j component of the mle times this expression right here. This expression right here is the absolute value of the j omle component minus lambda. Positive part where positive part is the argument of the positive part or 0, whichever one is bigger. So let's think a little bit what is this expression right here. This expression is the closed form solution to lasso regression in the case where the data is

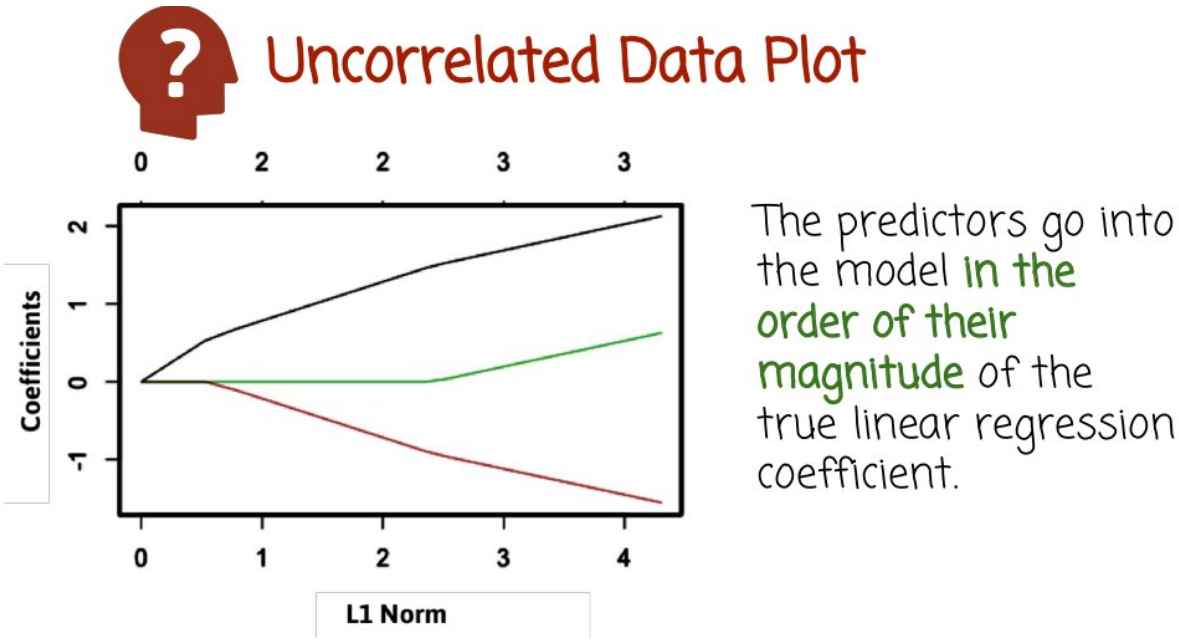
orthonormal.

So what is happening here? We're taking the mle coefficient and we are reducing it by lambda towards zero. And if we reduce it too much, meaning that it's going to go to the other side. If it was negative now it's positive, if it was positive now it's negative, then we just zero it out. The zeroing out is the sparsity encouraging part from before. So we take the absolute value, we reduce it by lambda. If we reduce it to something negative, we're just going to zero it out. If we do not reduce it to something smaller than zero, this expression right here is greater than zero. We need to give it the right sign. So we multiply it by the sign of the mle coefficient.

In summary, while ridge will simply reduce the mle or shrink it towards zero. Lasso will shrink it towards zero and then if it's shrunk sufficiently, so that it intersects the zero. It will just be set to zero and we will just basically remove or ignore that variable.

30 - Lasso Plots

So, let's see how we can plot some lasso parameters graph. We can do that with the r package glmnet, which we need to install it first, you can get some information on it from its website or from help functions.



Here is a Lasso graph in the uncorrelated case. So, what we see is you have some features that as we increase the penalty, they go down and then once they get to 0 they basically stay at 0. And once this one gets to 0 it stays at 0 and so on. If we think about it as going from the other direction, from the direction of very strong penalty into less penalty. Then we basically can think of coefficients entering the model. And they enter the model in the order of the magnitude of the linear regression coefficients.

31 - Mtcars and lasso Quiz

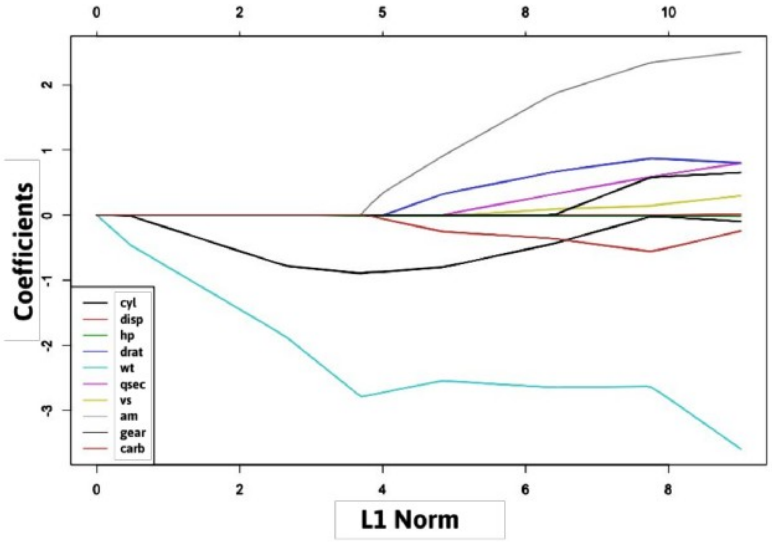
So here we see the graph of lasso coefficients, and they correspond to the mtcars dataset. Please, write the r commands needed to create this graph and the graph should be similar to the graph that you see here.

32 - Mtcars and lasso Quiz Solution

Here's one solution. We first installed the package glmnet and we bring it into scope and then we call the glmnet function. And we need to provide it with a data set, and we need to make sure we give it separate the labels. It knows what variable to predict. In this case, we chose the first component as the first feature as the target variable. And here's the code to generate the graph.

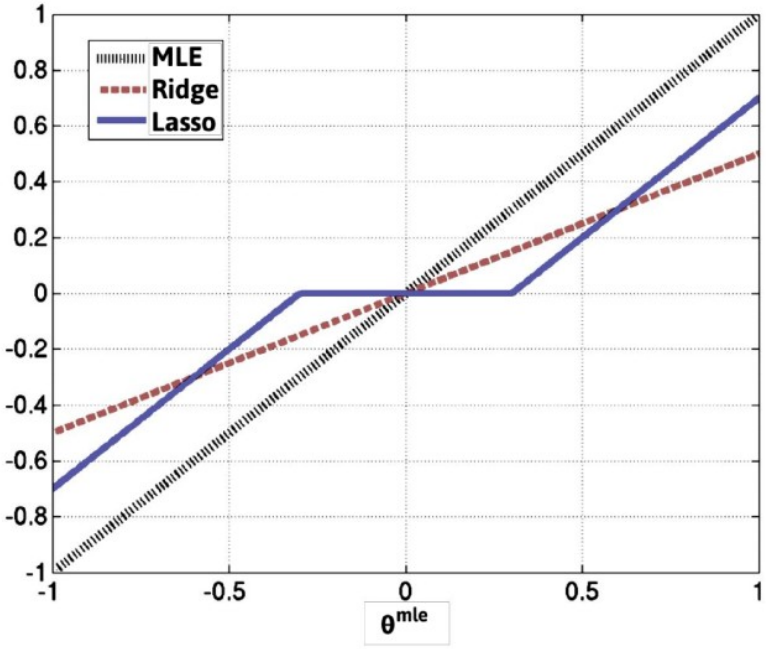
Mtcars and lasso Quiz

In the programming node, **write the commands needed to use the lasso model on the mtcars dataset.** Your plot should be similar to mine.



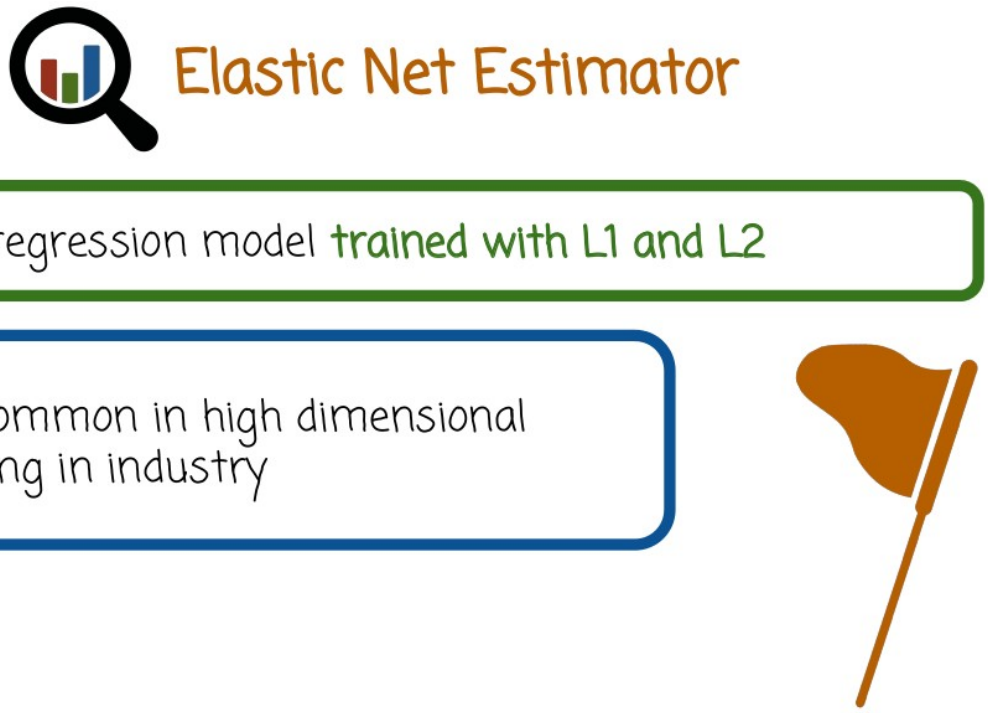
Mtcars and lasso Quiz

```
Install.packages("glmnet")
library(glmnet)

mtcarsLasso = glmnet(as.matrix(mtcars[-1]), mtcars
[,1])

plot(mtcarsLasso, col=1:12, lwd=2)
legend('bottomleft',legend=names(mtcars)[-1],col=1:
12,lty=1,lwd=1)
```

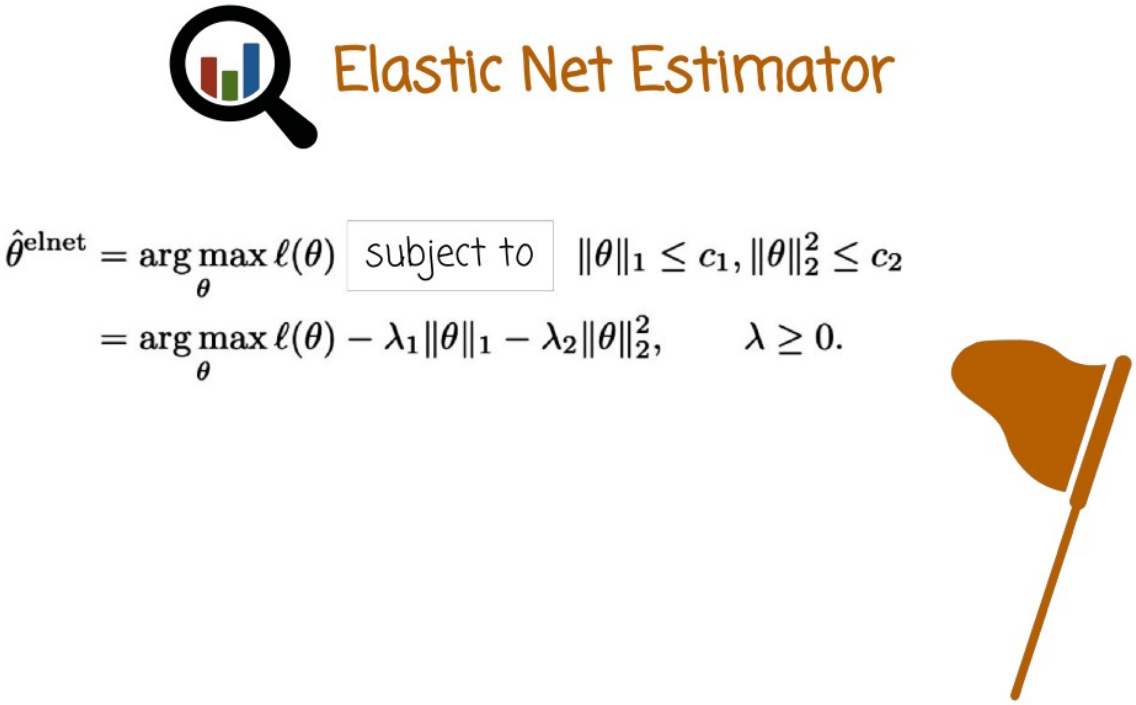


So if we look at the graphs of the MLE and the ridge and the lasso in the orthonormal case. That can give us some nice intuition about what's happening even in the non-orthonormal case. So in this graph we see the x axis mapping to the MLE parameter and the y axis mapping to the parameter of the MLE, the ridge or the lasso. And of course, MLE maps to the MLE using the identity or y equals x function. So we see here a straight line with a slope of one, because it just maps to itself.

What happens to the ridge? Again, this is only in the orthonormal case where we have closed form in the different dimensions decomposing to different optimization problems with the closed form solution. So in the case of the ridge, basically what it does is it maps the MLE into just a shrunk version of the MLE. Basically, if you remember similar to the James-Stein shrinkage is multiplied by some factor that's smaller than one. So we could easily see from the mathematical expression earlier. What's a little bit more interesting is the lasso graph where we look at. Again, this is for the orthonormal case, the lasso graph is a function of the MLE. And the shrinkage, what happens here is that the black line is just reduced. The shrinkage, rather than being multiplicative, the shrinkage is subtractive. So basically subtract lambda from the MLE value. And we keep subtracting, so the shrinkage is the difference between the black line and the blue line. And the shrinkage is the same until we get to a point where the shrinkage would've made the blue line goes beyond the line y equals 0 to the other side. And then we just zero it out. That kind of flat portion here, that's the region where the MLE will be zeroed out and the variable will be ignored. Here we get a symmetric situation in the case where theta is negative. So basically ridge shrinkage is multiplicative. You see here by the shrinkage going down and down and down and down as we get closer to zero. The shrinkage amount changes, because it's multiplicative shrinkage whereas lasso is more subtractive shrinkage. We reduce the MLE parameter by some amount until we get to a place where the reduction would make it negative and then we just zero it out. And similarly from the negative side. So this is not generally, that's just for the orthonormal case. But it helps understand the difference between ridge and lasso and why lasso will provide more sparse solutions.

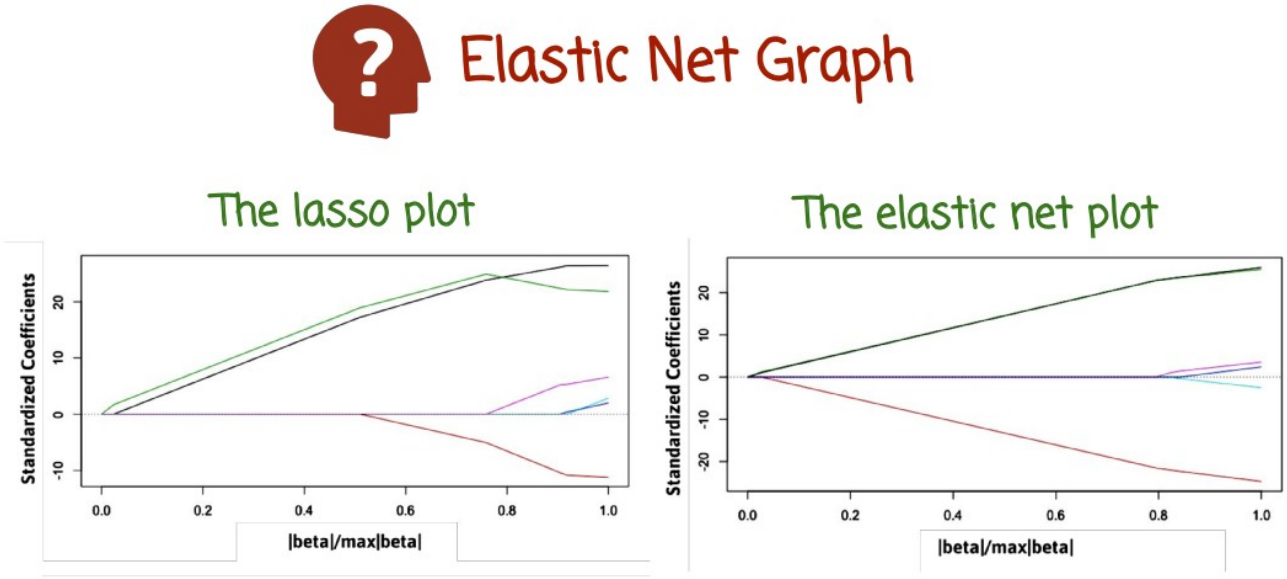


So the last regularization method that we'll see is called elastic net, which is basically a combination of L1 regularization and L2 regularization, or ridge and lasso. It's very common in high dimensional modeling. It combines benefits of L1 and L2.



And here is the formulation. We maximize the likelihood subject to two different constraints. One constraint on the L1 norm, the other one is on the L2 norm. We form the Lagrangian, because we don't want to solve this optimization problem. And we get the dual optimization problem and here we have two penalties. And we have two parameters, lambda1 and lambda2, both of them greater than 0. And we can play with their values in order to get different types of regularizations, stronger or weaker, and different amounts of sparsity. If we want more sparsity, we can increase lambda1 compared to lambda2. If we don't want sparsity that much, we can make lambda1 smaller and lambda2 larger. In general, since it includes both L1 and L2 as special cases, if you set lambda1 to zero, you get ridge. If you set lambda2 to zero, you get Lasso. This is considered more general, and it does not have much more computation load beyond the lasso solution. So basically, if you do the lasso, you might as well do elastic net. Because it gives you a little bit more flexibility, because you have two regularization parameters and you can control the amount of shrinkage that is sparsity-inducing and shrinkage that is not sparsity-inducing.

35 - Elastic Net Graph



Let's see how the elastic net parameter graph differs from lasso. And so here we have again in the x-axis the amount of regularization, and on the y-axis the value of the coefficient. So, here in the case of lasso, we see the coefficients being shrunk to zero and then they remain zero. The corresponding same data set elastic net, we see something between lasso and ridge. And so of course it depends on the amount of L1 and L2 regularization, but you still see some coefficients being sent to zero or getting some sparsity but in this case, these two coefficients do not become zero until the very end. So the sparsity is, relatively speaking of course, again depend on the magnitude of L1 and L2 regularization, but relatively speaking sparsity may not be as aggressive but you do get more reduction in magnitude. But you do get some sparsity in contrast to ridge that has no sparsity whatsoever.

36 - Lesson Summary

In this lesson, we saw that overfitting is a very significant problem in high dimensions, as it can lead to

low quality models. We saw several ways of handling overfitting in the case of linear and logistic regression. Such high dimensional regularized linear models are frequently in use all over today's high tech industry.

37 - Course Summary

We've completed the video portion of the course. By now, you should be comfortable with R, data analysis, data visualization, and data modeling.

The project should help you get a practical take on understanding of these concepts and how to apply them effectively in the real world. As you move beyond this course and learn additional topics related to data visualization and analysis, it is important to understand that academic was theoretical knowledge can only get you so far. You must get your hands dirty and analyze, visualize and model different types of data and experiment with different approaches. Good luck in your road ahead in this exciting and hugely important field.