CS M148 –

Data Science Fundamentals

Lecture #6: Inference cont. & Regularization

Baharan Mirzasoleiman

UCLA Computer Science

Announcements

HW 1 is posted

- Due Wed Jan 26, 2pm

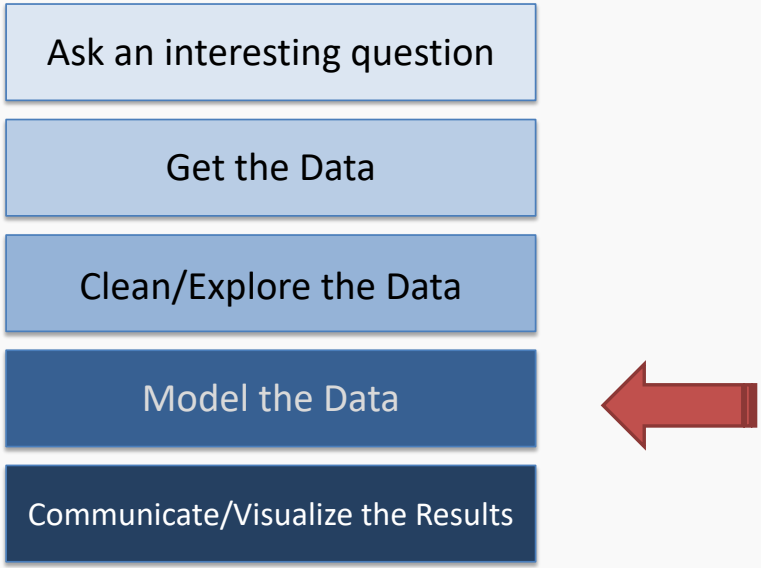
Project 2 will be posted

- Will be released on Jan 26
- Due Wed Feb 9, 2pm

Let’s quickly review what we saw last time

Ready to Model the Data!

The Data Science Process



Model Selection

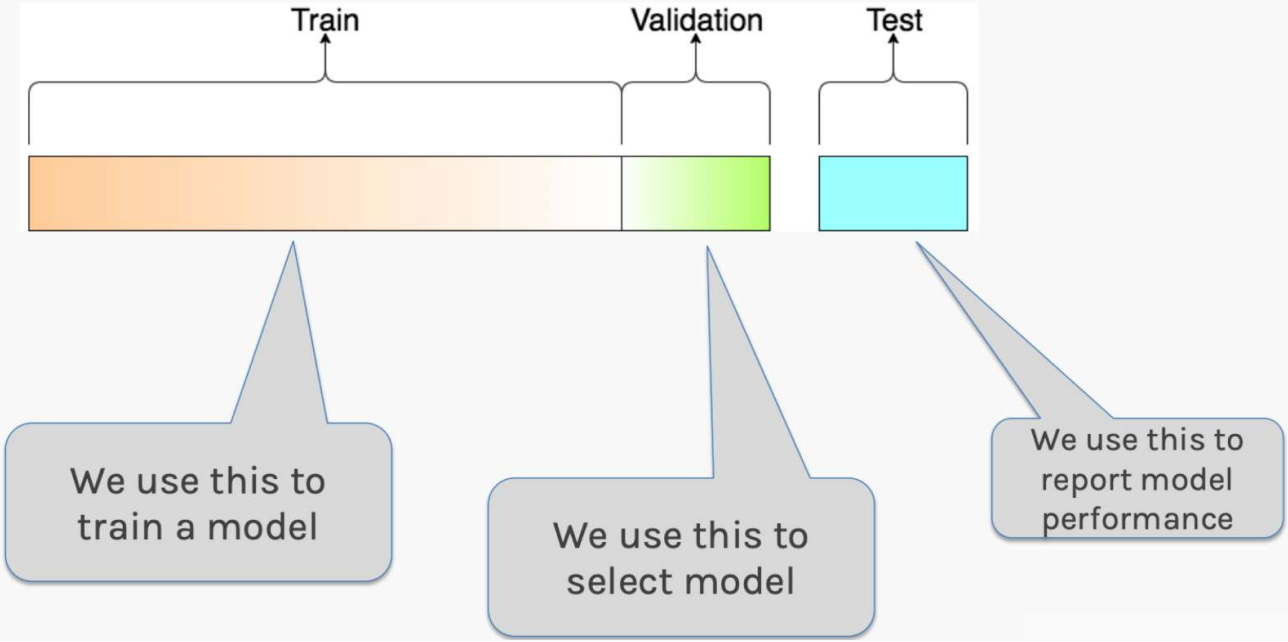
Generalization Error

We know to evaluate the model on both train and test data, because models that do well on training data may do poorly on new data (overfitting).

The ability of models to do well on new data is called generalization.

The goal of model selection is to choose the model that generalizes the best.

Train-Validation-Test



Stepwise Variable Selection: Forward method

In **forward selection**, we find an 'optimal' set of predictors by iteratively building up our set.

1. Start with the empty set P0, construct the null model M0.

2. For $k = 1, \dots, J$:

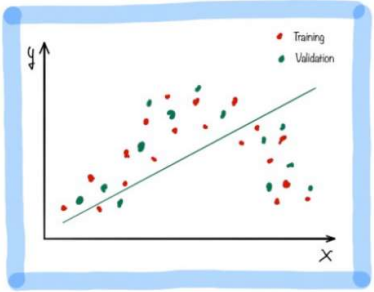
2.1 Let M_{k-1} be the model constructed from the best set of $k-1$ predictors, P_{k-1}.

2.2 Select the predictor X_{n_k}, not in P_{k-1}, so that the model constructed from $P_k = X_{n_k} \cup P_{k-1}$ optimizes a fixed metric (this can be p-value, F-stat; validation MSE, R^2, or AIC/BIC on training set).

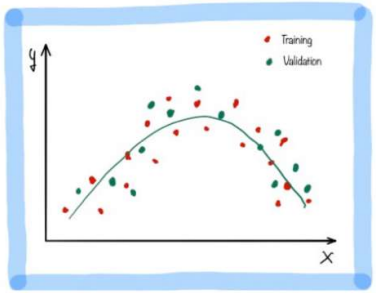
2.3 Let M_k denote the model constructed from the optimal P_k.

3. Select the model M amongst $\{M_0, M_1, \dots, M_J\}$ that optimizes a fixed metric (this can be validation MSE, R^2; or AIC/BIC on training set)

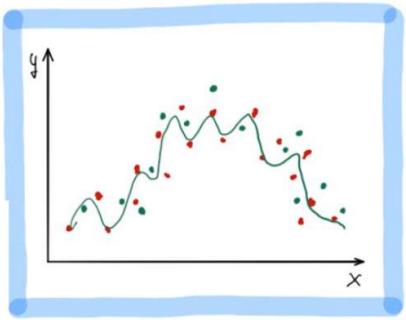
Underfitting: train and validation error is high.

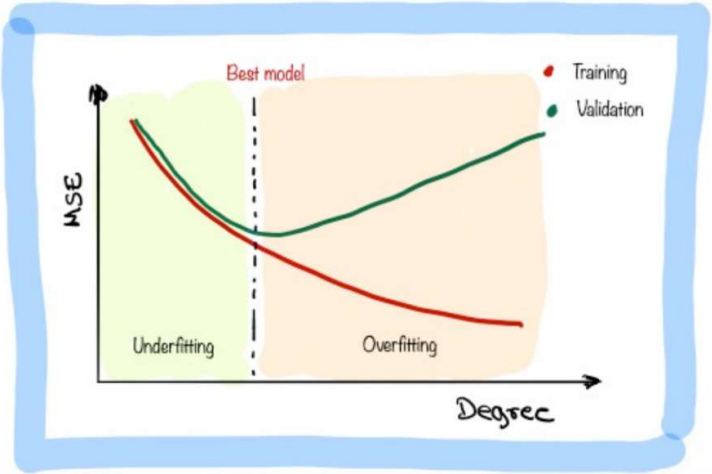


Best model: validation error is minimum.



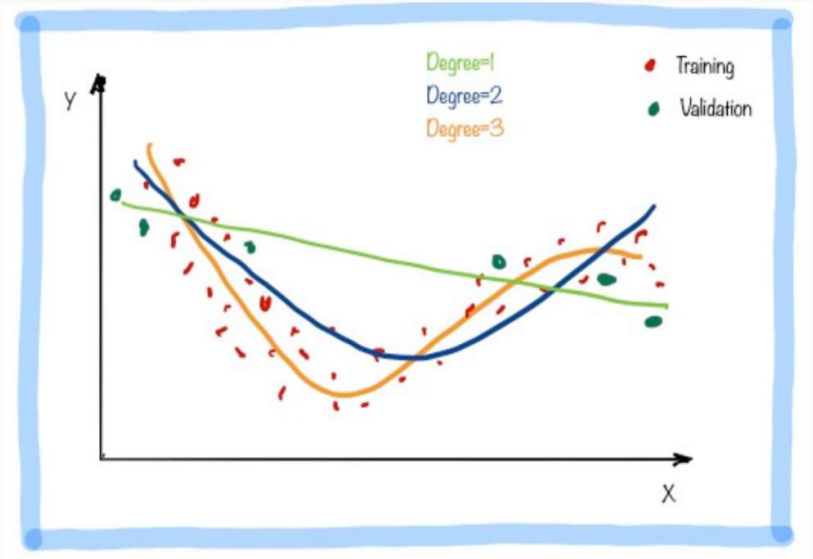
Overfitting: train error is low, validation error is high.





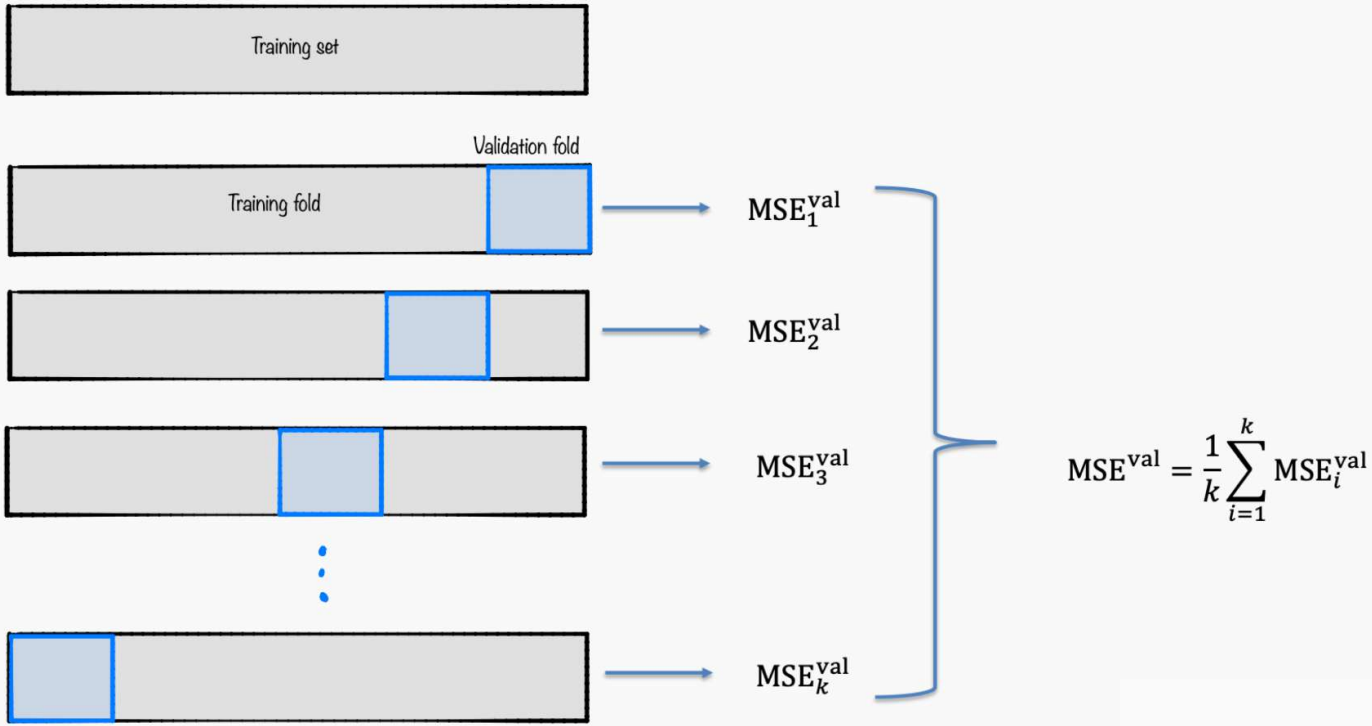
Cross Validation: Motivation

Using a single validation set to select amongst multiple models can be problematic - **there is the possibility of overfitting to the validation set**



It is obvious that degree=3 is the correct model but the validation set by chance favors the linear model.

Cross Validation



K-Fold Cross Validation

Given a data set $\{X_1, \dots, X_n\}$, where each $\{X_1, \dots, X_n\}$ contains *J* features.

To ensure that every observation in the dataset is included in at least one training set and at least one validation set we use the **K-fold validation**:

- split the data into *K* uniformly sized chunks, $\{C_1, \dots, C_K\}$
- we create *K* number of training/validation splits, using one of the *K* chunks for validation and the rest for training.

We fit the model on each training set, denoted $\hat{f}_{C_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{C_{-i}}(C_i)$. The ***cross validation is the performance*** of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{K}\sum_{i=1}^{K} L(\hat{f}_{C_{-i}}(C_i))$$

where *L* is a loss function.

Leave-One-Out

*Or using the **leave one out*** method:

- validation set: $\{X_i\}$
- training set: $X_{-i} = \{X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n\}$

for $i = 1, \dots, n$:

We fit the model on each training set, denoted $\hat{f}_{X_{-i}}$, and evaluate it on the corresponding validation set, $\hat{f}_{X_{-i}}(X_i)$.

The ***cross validation score*** is the performance of the model averaged across all validation sets:

$$CV(\text{Model}) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{f}_{X_{-i}}(X_i))$$

where *L* is a loss function.

Inference in Linear Regression

Uncertainty in estimating the linear regression coefficients

Outline

Assessing the Accuracy of the Coefficient Estimates

- Bootstrapping and confidence intervals

Evaluating Significance of Predictors

- Does the outcome depend on the predictors?
- Hypothesis testing

How well do we know $\hat{f}$

- The confidence intervals of $\hat{f}$

How reliable are the model interpretation

Suppose our model for advertising is:

$$y = 1.01x + 120$$

Where y is the sales in 1000$, x is the TV budget.

Interpretation: for every dollar invested in advertising, you get 1.01 back in sales, which is 1% net increase.

But how certain are we in our estimation of the coefficient 1.01?

Why aren't we certain?

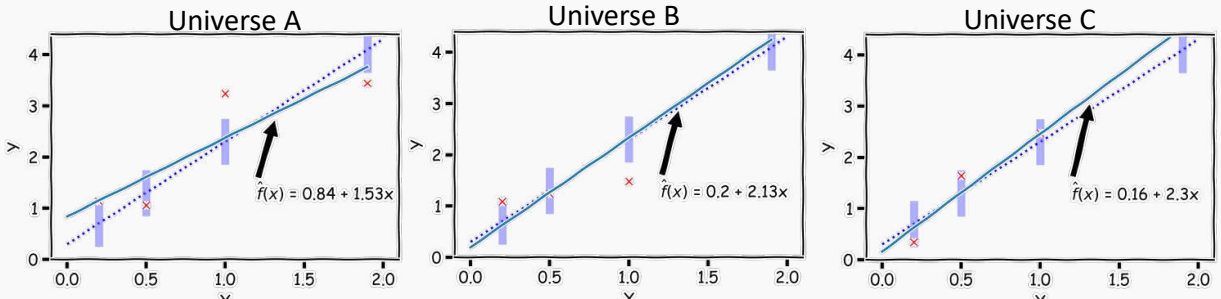
Confidence intervals for the predictors estimates (cont)

So if we have one set of measurements of $\{X, Y\}$, our estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are just for this particular realization.

Question: If this is just one realization of reality, how do we know the truth? How do we deal with this conundrum?

Our data is just a particular sample of the population

Imagine (magic realism) we have parallel universes, and we repeat this experiment on each of the other universes.



Bootstrapping and Confidence Intervals

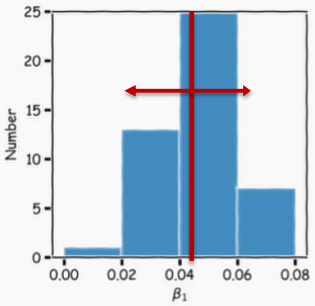
Bootstrapping for Estimating Sampling Error

Definition

Bootstrapping is the practice of estimating properties of an estimator by measuring those properties by, for example, sampling from the observed data.

For example, we can compute $\hat{\beta}_0$ and $\hat{\beta}_1$ multiple times by randomly sampling from our data set. We then use the variance of our multiple estimates to approximate the true variance of $\hat{\beta}_0$ and $\hat{\beta}_1$.

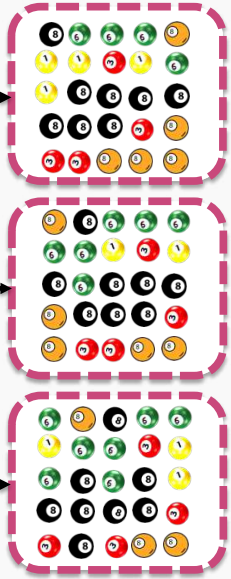
Bootstrap



Size N

Dataset

Sample with replacement



train

Model 1: $\hat{y} = \hat{\beta}_0^{(1)} + \hat{\beta}_1^{(1)} x$

train

Model 2: $\hat{y} = \hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)} x$

train

Model s: $\hat{y} = \hat{\beta}_0^{(s)} + \hat{\beta}_1^{(s)} x$

$$\mu_{\hat{\beta}} = \frac{1}{s}\sum_{i=1}^{s} \hat{\beta}^{(i)}$$

$$\sigma_{\hat{\beta}} = \sqrt{\frac{1}{s}\sum_{i=1}^{s} \left(\hat{\beta}^{(i)} - \bar{\beta}\right)^2}$$

Confidence intervals for the predictors estimates: **Standard Errors**

We can empirically estimate the standard deviations $\sigma_{\hat{\beta}}$ which are called the **standard errors,** $SE(\hat{\beta}_0), SE(\hat{\beta}_1)$ through bootstrapping.

Alternatively:

If we know the variance σ_ϵ^2 of the noise ϵ, we can compute $SE(\hat{\beta}_0), SE(\hat{\beta}_1)$ analytically using the formula below (no need to bootstrap):

$$SE\left(\hat{\beta}_0\right) = \sigma_\epsilon \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_i (x_i - \bar{x})^2}}$$

$$SE\left(\hat{\beta}_1\right) = \frac{\sigma_\epsilon}{\sqrt{\sum_i (x_i - \bar{x})^2}}$$

Where n is the number of observations

$\bar{x}$ is the mean value of the predictor.

Standard Errors

More data: $n \uparrow$ and $\sum_i (x_i - \bar{x})^2 \uparrow \Longrightarrow SE \downarrow$

Larger coverage: $var(x)$ or $\sum_i (x_i - \bar{x})^2 \uparrow \Longrightarrow SE \downarrow$

Better data: $\sigma_\epsilon^2 \downarrow \Rightarrow SE \downarrow$

$$SE\left(\hat{\beta}_0\right) = \sigma_\epsilon \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum_i (x_i - \bar{x})^2}}$$

$$SE\left(\hat{\beta}_1\right) = \frac{\sigma(\epsilon)}{\sqrt{\sum_i (x_i - \bar{x})^2}}$$

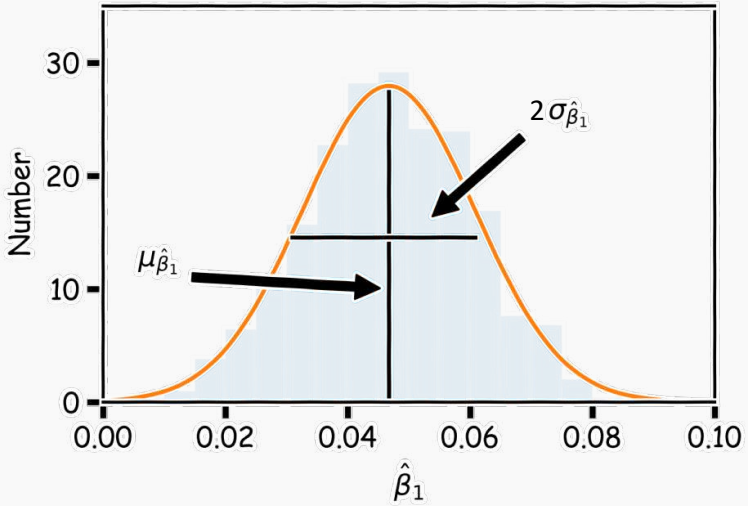
Better model: $(\hat{f} - y_i) \downarrow \Longrightarrow \sigma_\epsilon \downarrow \Longrightarrow SE \downarrow$

$$\sigma(\epsilon) = \sqrt{\sum \frac{\left(\hat{f}(x) - y_i\right)^2}{n - 2}}$$

Question: What happens to the $\widehat{\beta_0}$, $\widehat{\beta_1}$ under these scenarios?

Confidence intervals for the predictors estimates (cont)

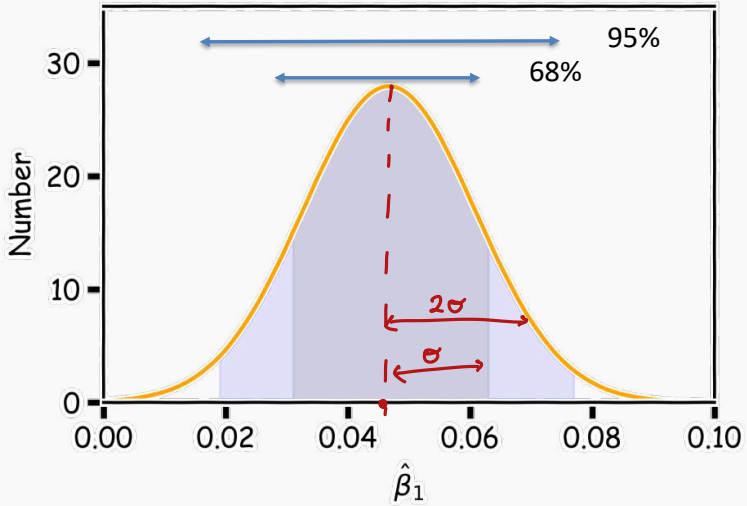
We can now estimate the mean and standard deviation of the estimates of $\hat{\beta}_0, \hat{\beta}_1$.



Confidence intervals for the predictors estimates (cont)

The standard errors give us a sense of our uncertainty over our estimates.

Typically we express this uncertainty as a 95% confidence interval, which is the range of values such that the **true** value of β_1 is contained in this interval with *95%* percent probability.



$$CI_{\hat{\beta}} = (\hat{\beta} - 2\sigma_{\hat{\beta}}, \hat{\beta} + 2\sigma_{\hat{\beta}})$$

Estimating Significance of Predictors
Hypothesis testing

How reliable are the model interpretation

Suppose our model for advertising is:

$$y = 1.01x + 120$$

Where y is the sales in $1000, x is the TV budget.

Interpretation: for every dollar invested in advertising gets you 1.01 back in sales, which is 1% net increase.

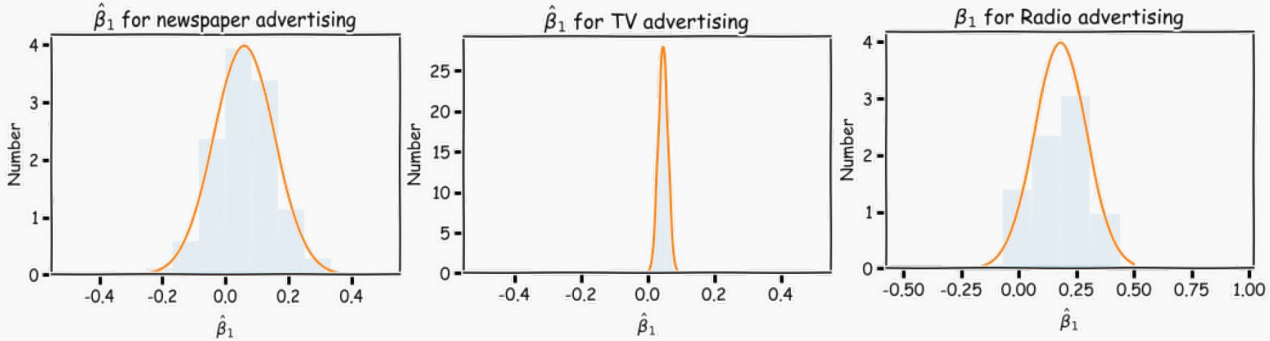
But how certain are we in our estimation of the coefficient 1.01?

Now you know how certain you are in your estimates, will you want to change your answer?

Feature importance

Now we know how to generate these distributions we are ready to answer ***two important questions:***

A. Which predictors are most important?
B. And which of them really affect the outcome?



$\mu_{\beta_1} = 0.03$
σ_{β_1}=0.13

$\mu_{\beta_1} = 0.033$
σ_{β_1}=0.01

$\mu_{\beta_1} = 0.23$
σ_{β_1}=0.25

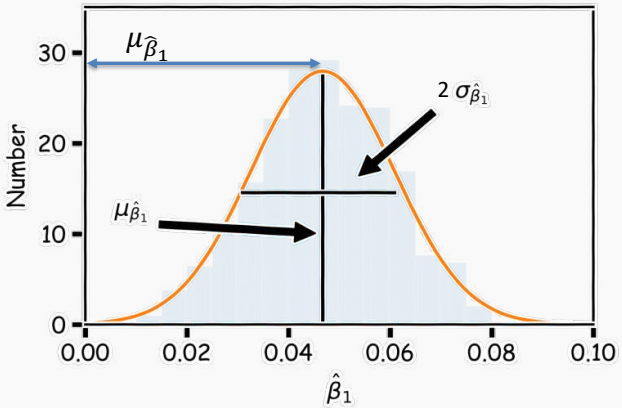
Feature Importance

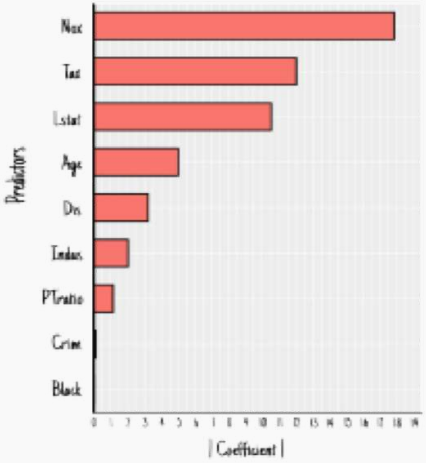
To incorporate the coefficients' uncertainty, we need to determine whether the estimates of $\beta's$ are sufficiently far from zero.

To do so, we define a new metric, which we call t-test statistic:

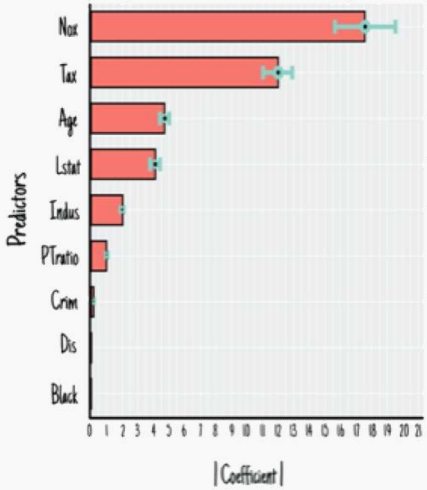
$$t = \frac{\mu_{\widehat{\beta}_1}}{\sigma_{\widehat{\beta}_1}}$$

which measures the distance from zero in units of standard deviation.

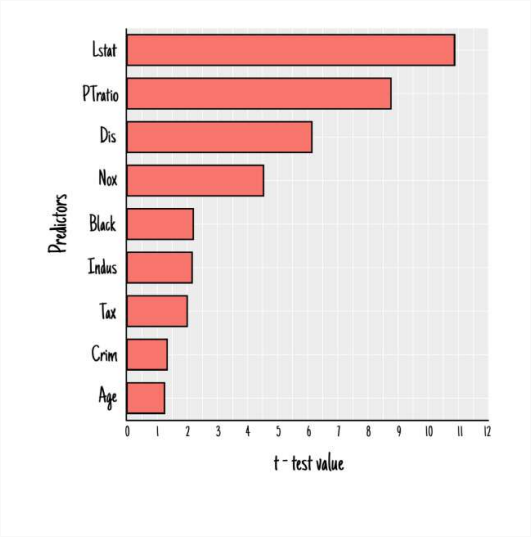




Feature importance base on the absolute value of the coefficients.



Feature importance base on the absolute value of the coefficients over multiple bootstraps and includes the uncertainty of the coefficients.



Feature importance base on t-test. Notice the rank of the importance has changed.

- we want a large t-value

Feature Importance

Because a predictor is ranked as the most important, it does not necessarily mean that the outcome depends on that predictor.

How do we assess if there is a true relationship between outcome and predictors?

As with R-squared, we should compare its significance (t-test) to the equivalent measure from a dataset where we know that there is no relationship between predictors and outcome.

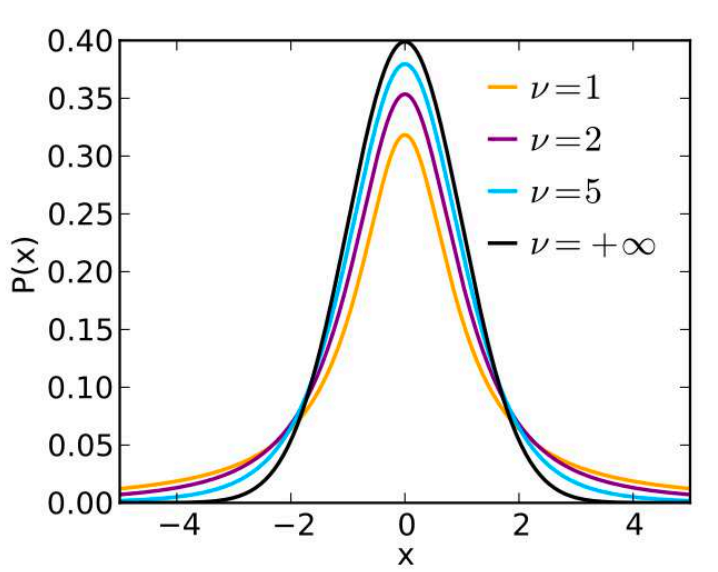
We are sure that there will be no such relationship in data that are randomly generated. Therefore, we want to compare the t-test of the predictors from our model with t-test values calculated using random data.

1. For n random datasets fit n models.
2. Generate distributions for all predictors and calculate the means and standard errors $(\mu_{\hat{\beta}}, \sigma_{\hat{\beta}})$.
3. Calculate the t-tests.

Repeat and create a probability density function (pdf) for all the t-tests.

It turns out we do not have to do this, because this is a known distribution called student-t distribution.

↳ t^R



Student-t distribution, where ν is the degrees of freedom (number of data points minus number of predictors).

To learn more about why student-t, what are degrees of freedom and more details see https://en.wikipedia.org/wiki/Student%27s_t-test

P-value

To compare the t-test values of the predictors from our model, $|t^*|$, with the t-tests, calculated using random data, $|t^R|$, we estimate the probability of observing $|t^R| \geq |t^*|$.

We call this probability the p-value.

$$p - value = P(|t^R| \geq |t^*|)$$

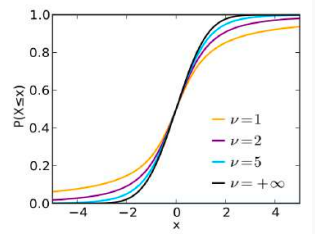
if $|t^*| \geq |t^R|$, then it means our predictor is significant

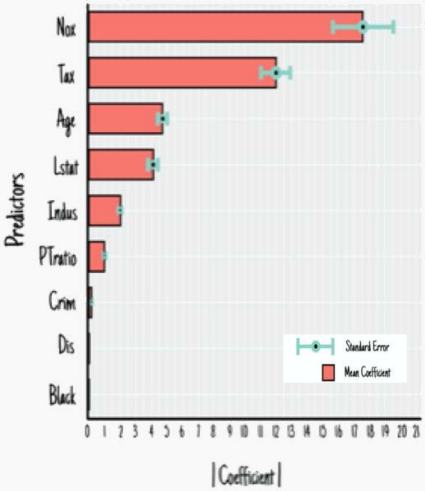
small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance.

It is common to use p-value<0.05 as the threshold for significance.

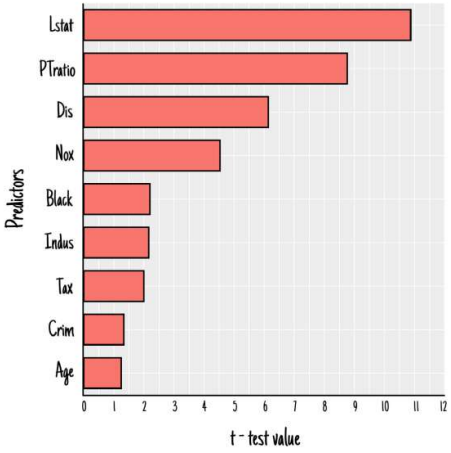
To calculate the p-value we use the cumulative distribution function (CDF) of the student-t.

`stats model` a python library has a build-in function `stats.t.cdf()` which can be used to calculate this.

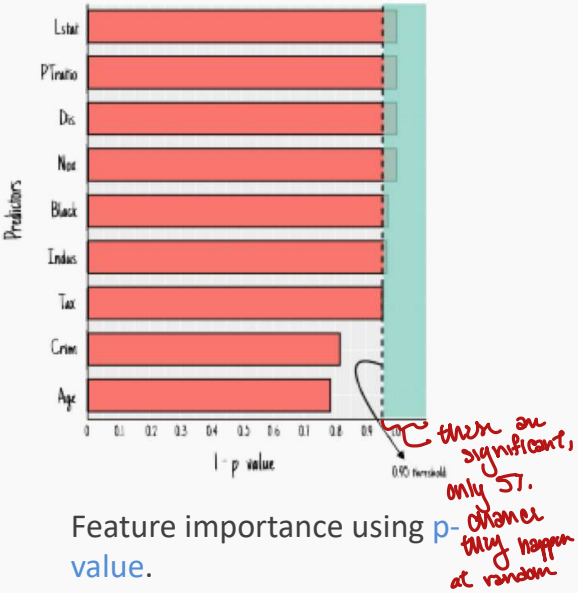




Feature importance based on the absolute value of the coefficients over multiple bootstraps and includes the coefficients' uncertainty.



Feature importance based on t-test. Notice the rank of the importance has changed.



Feature importance using p-value.

Hypothesis Testing

Hypothesis testing is a formal process through which we evaluate the validity of a statistical hypothesis by considering evidence **for** or against the hypothesis gathered by **random sampling** of the data.

1. State the hypotheses, typically a **null hypothesis**, H_0 and an **alternative hypothesis**, H_1, that is the negation of the former.
2. Choose a type of analysis, i.e. how to use sample data to evaluate the null hypothesis. Typically this involves choosing a single test statistic.
3. **Sample** data and compute the test statistic.
4. Use the value of the test statistic to either reject or not reject the null hypothesis.

Hypothesis testing

1. State Hypothesis:

Null hypothesis:

H_0: There is no relation between *X* and *Y*

The alternative:

H_a: There is some relation between *X* and *Y*

2. **Choose test statistics**

t-test

3. Sample:

Using bootstrap we can estimate $\hat{\beta}_1's$, and $\mu_{\hat{\beta}_1}$ and $\sigma_{\hat{\beta}_1}$ and the t-test.

Hypothesis testing

4. **Reject or not reject the hypothesis:**

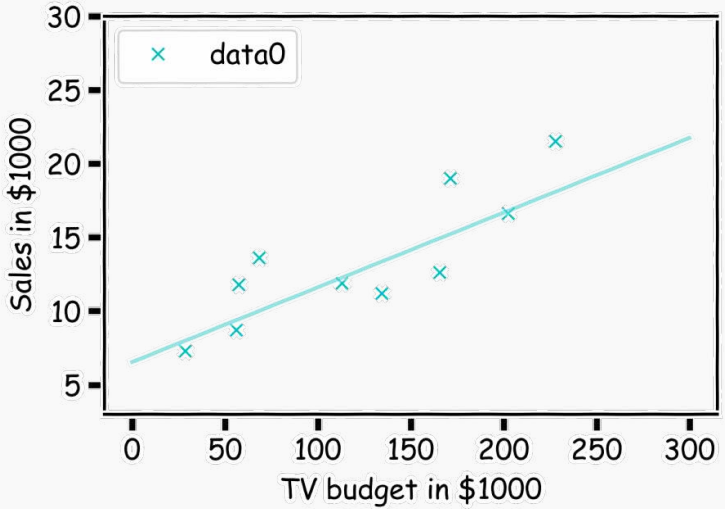
   We compute ***p-value ,*** the probability of observing any value equal to $|t|$ or larger, from random data.

   p-value < p-value-threshold we reject the null.

Prediction Intervals

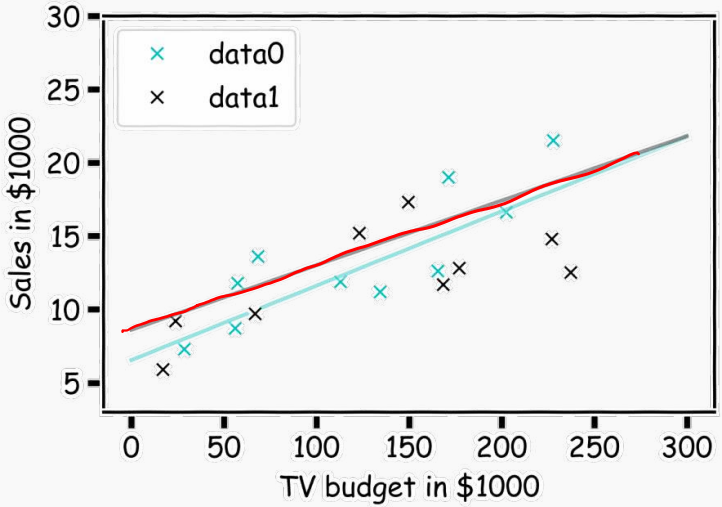
How well do we know $\hat{f}$?

Our confidence in f is directly connected with our confidence in βs. For each bootstrap sample, we have one β, which we can use to determine the model, $f(x) = X\beta$.



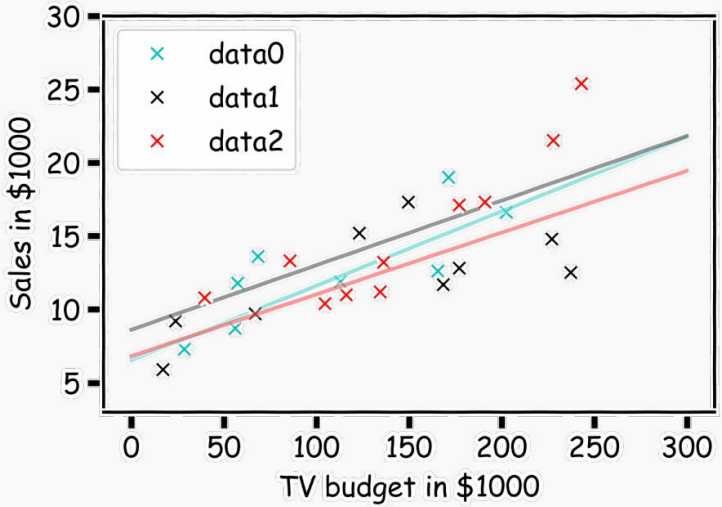
How well do we know $\hat{f}$?

Here we show two difference models predictions given the fitted coefficients.



How well do we know $\hat{f}$?

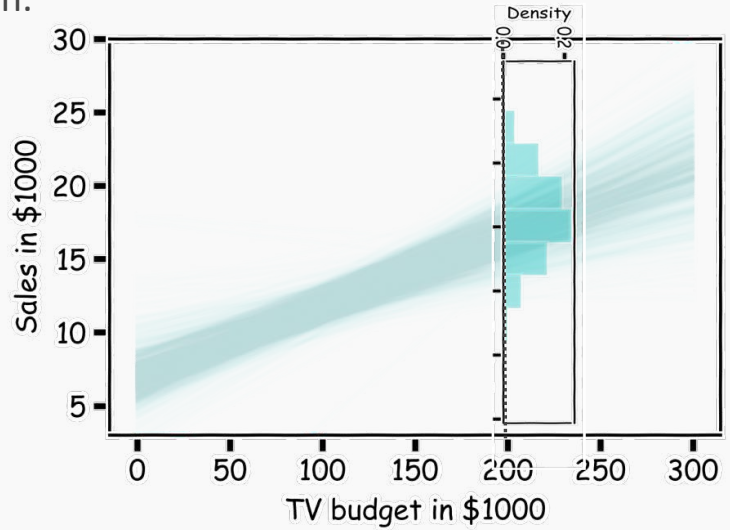
There is one such regression line for every bootstrapped sample.



How well do we know $\hat{f}$?

Below we show all regression lines for a thousand of such bootstrapped samples.

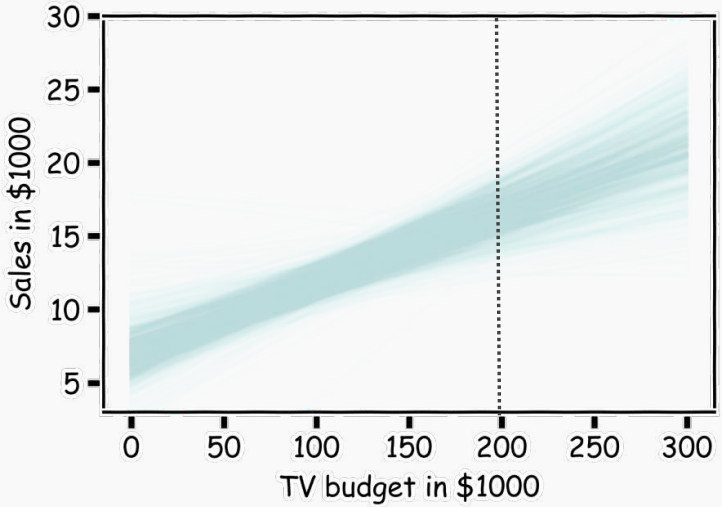
For a given x, we examine the distribution of $\hat{f}$, and determine the mean and standard deviation.

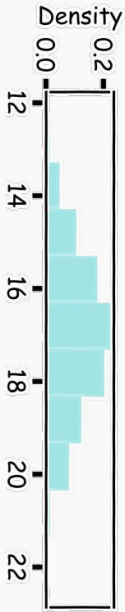


How well do we know $\hat{f}$?

Below we show all regression lines for a thousand of such bootstrapped samples.

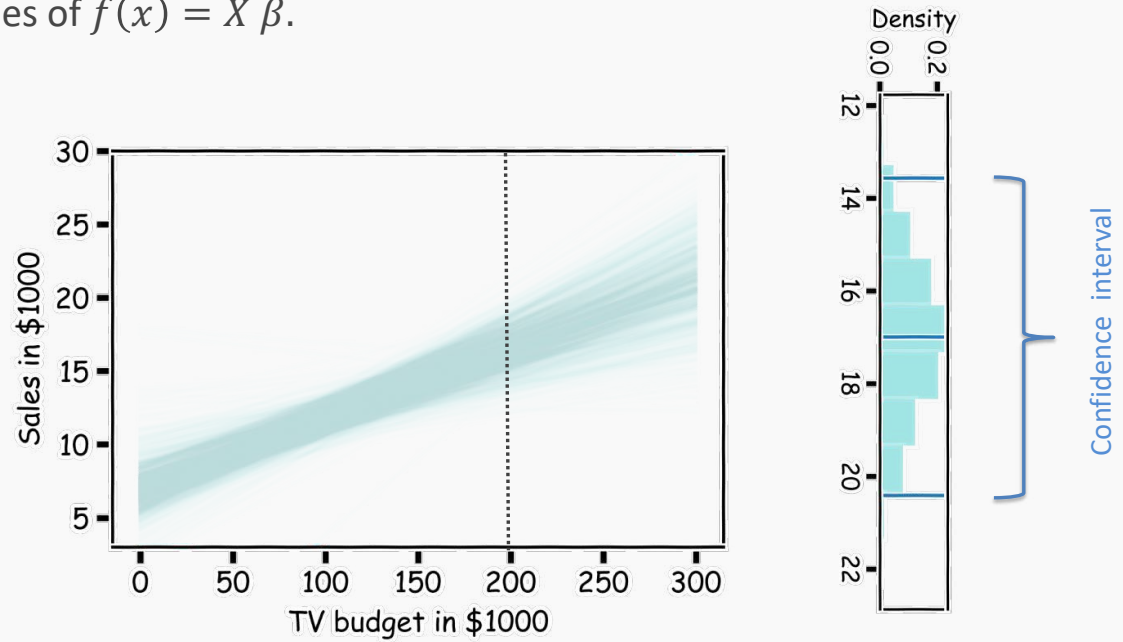
For a given x, we examine the distribution of $\hat{f}$, and determine the mean and standard deviation.





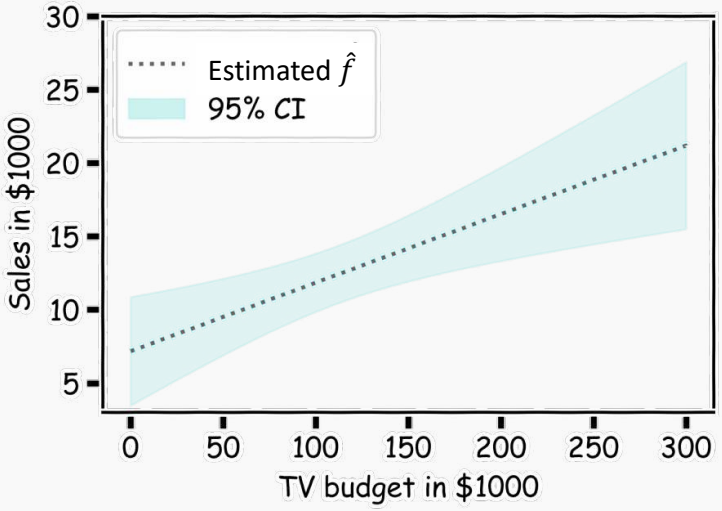
How well do we know $\hat{f}$?

We determine the confidence interval of $\hat{f}$ by selecting the region that contains 95% of the samples of $\hat{f}(x) = X\,\hat{\beta}$.



How well do we know $\hat{f}$?

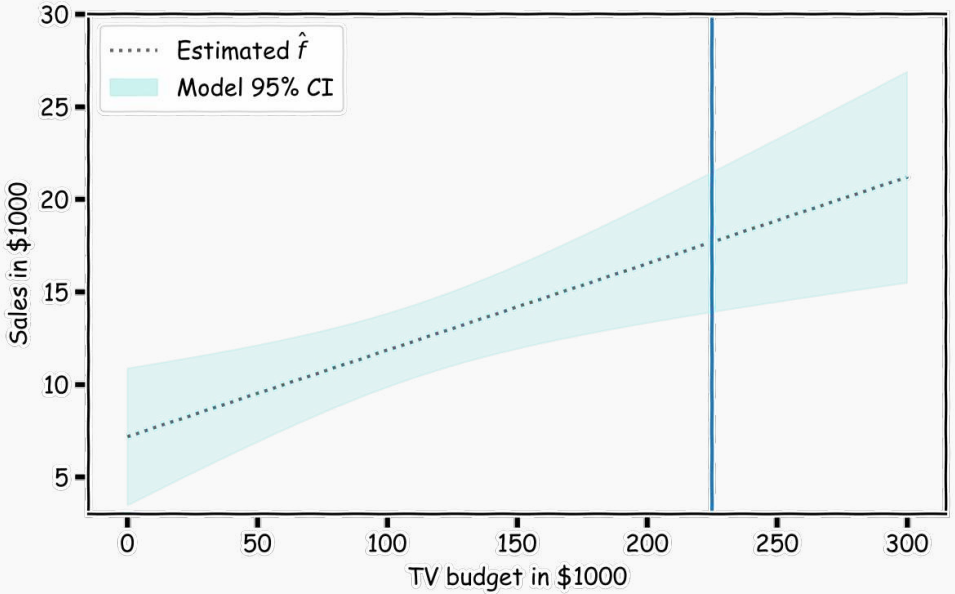
For every x, we calculate the mean of the models, $\widehat{\mu_f}$ (shown with dotted line) and the 95% CI of those models (shaded area).



Confidence in predicting $\hat{y}$

Even if we knew $f(x)$ —the response value cannot be predicted perfectly because of the random error in the model (irreducible error).

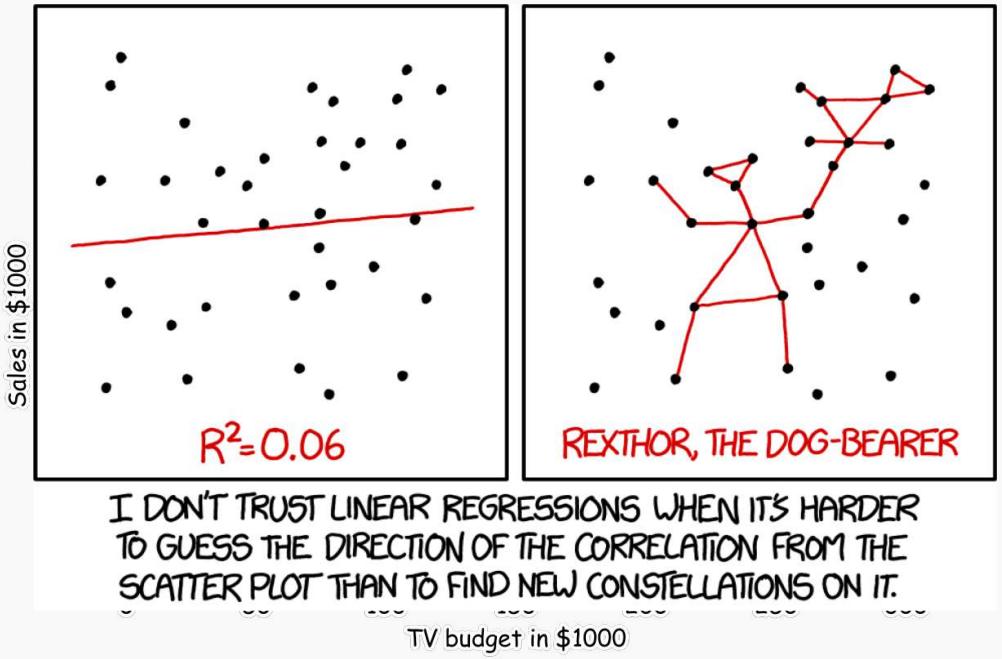
How much will Y vary from $\hat{Y}$? We use prediction intervals to answer this question.



Confidence in predicting $\hat{y}$

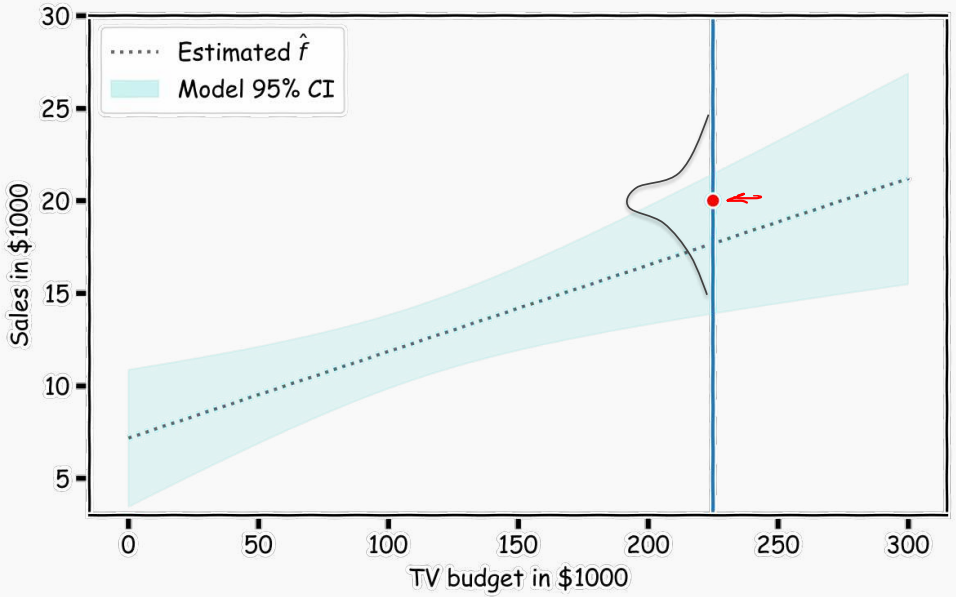
Even if we knew $f(x)$ —the response value cannot be predicted perfectly because of the random error in the model (irreducible error).

How much will Y vary from $\hat{Y}$? We use prediction intervals to answer this question.



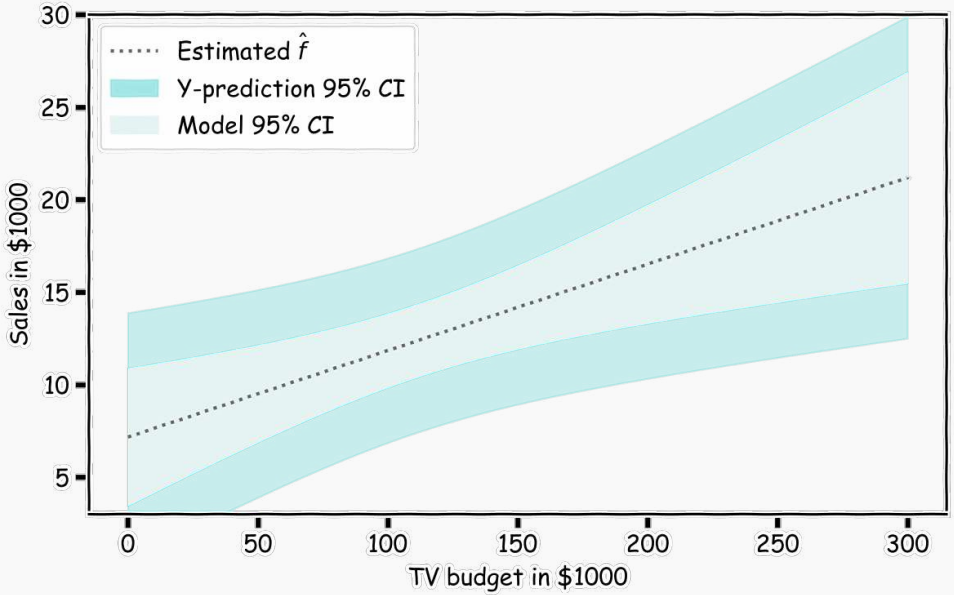
Confidence in predicting $\hat{y}$

- for a given x, we have a distribution of models $f(x)$
- for each of these $f(x)$, the prediction for $y \sim N(f(x), \sigma_\epsilon)$



Confidence in predicting $\hat{y}$

- for a given x, we have a distribution of models $f(x)$
- for each of these $f(x)$, the prediction for $y \sim N(f(x), \sigma_\epsilon)$
- The prediction confidence intervals are then …



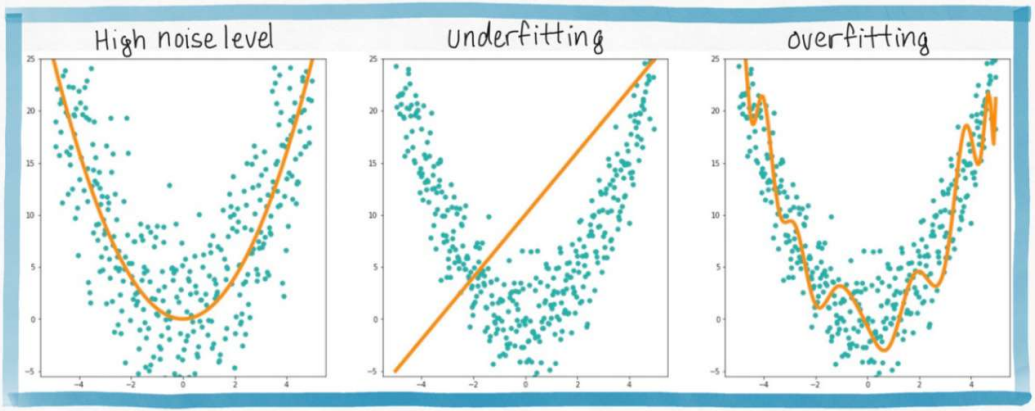
Regularization error, bias vs variance

Test Error and Generalization

We know to evaluate models on both train and test data because models can do well on training data but do poorly on new data.

When models do well on new data is called generalization.

There are at least three ways a model can have a high test error.



Irreducible and Reducible Errors

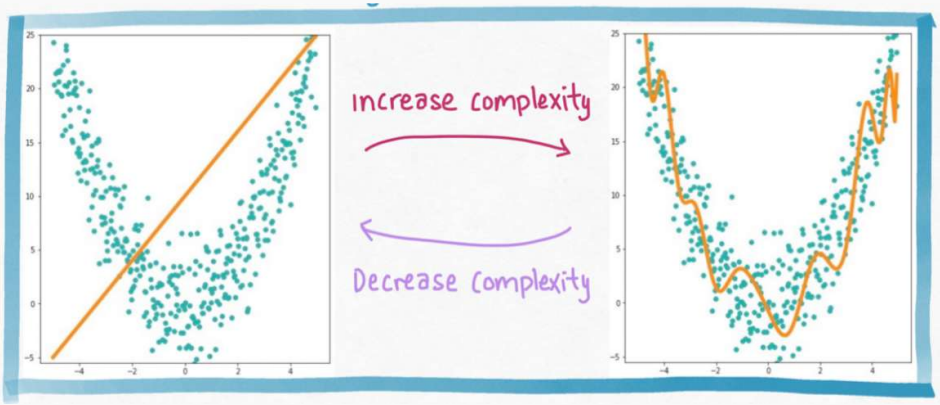
We distinguished the contributions of noise to the generalization error:

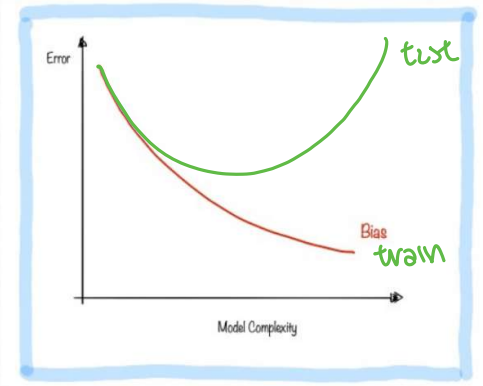
Irreducible error: we can’t do anything to decrease error due to noise.

Reducible error: we can decrease error due to overfitting and underfitting by improving the model.

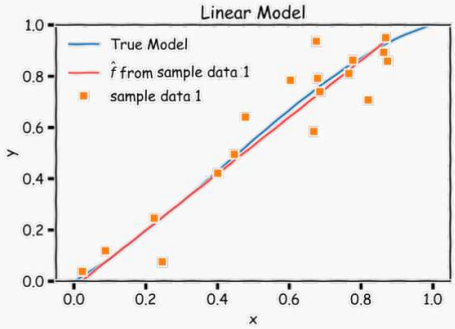
The Bias-Variance: Bias

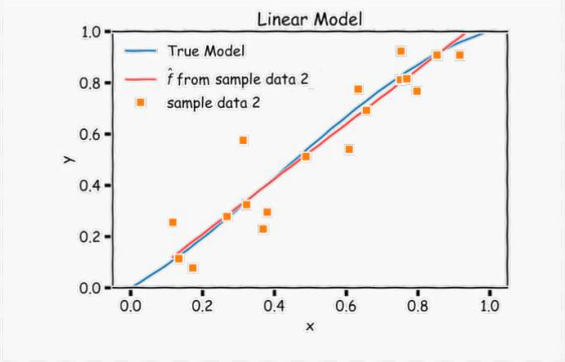
Reducible error comes from either underfitting or overfitting. There is a trade-off between the two sources of errors:

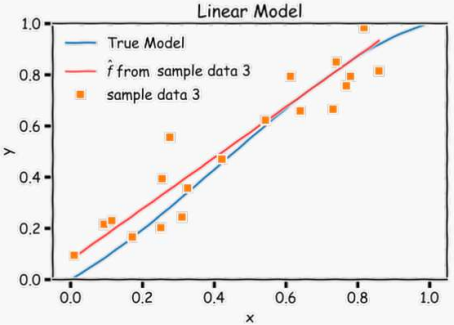


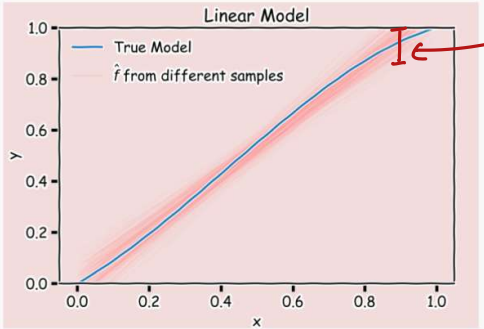


Bias vs Variance: Variance



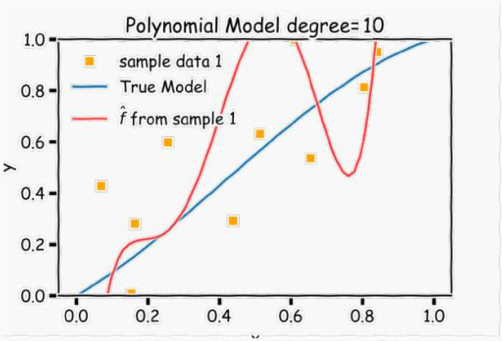


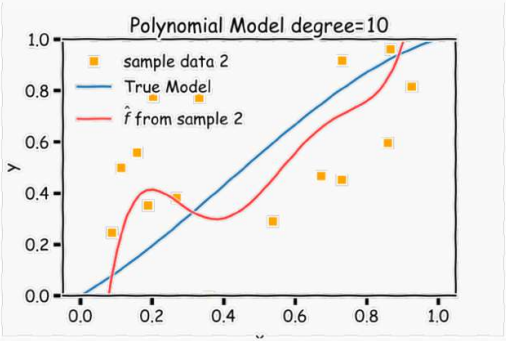


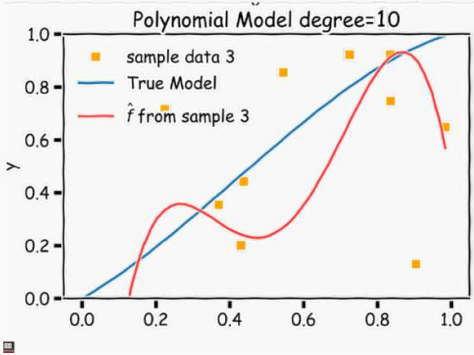


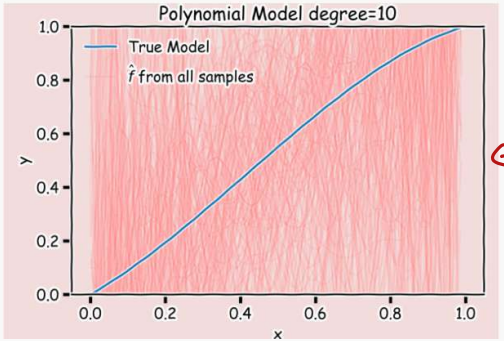
smaller variance

Bias vs Variance







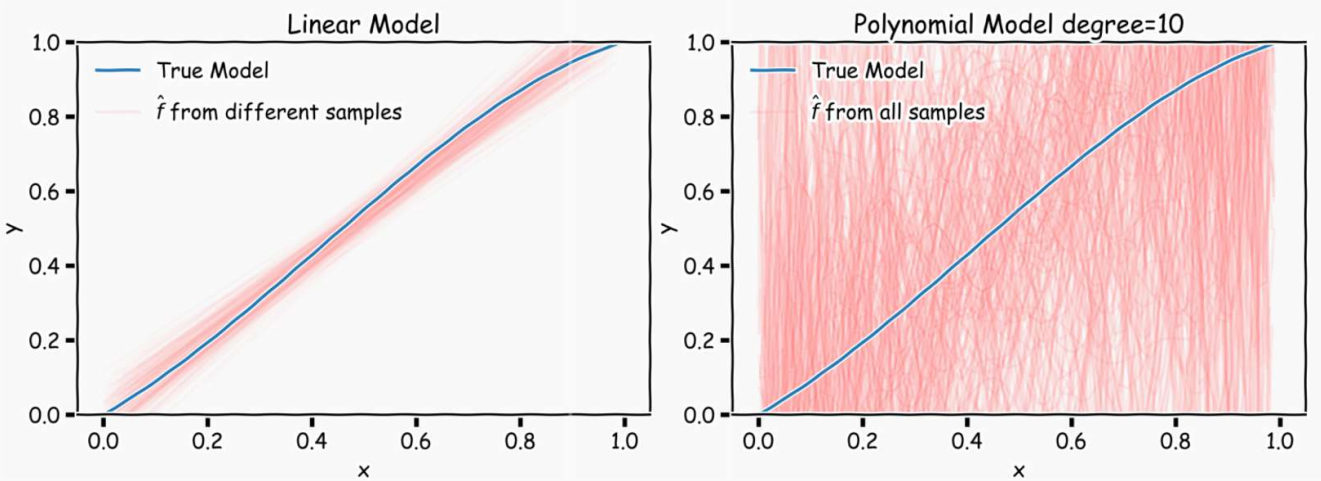


← large variance

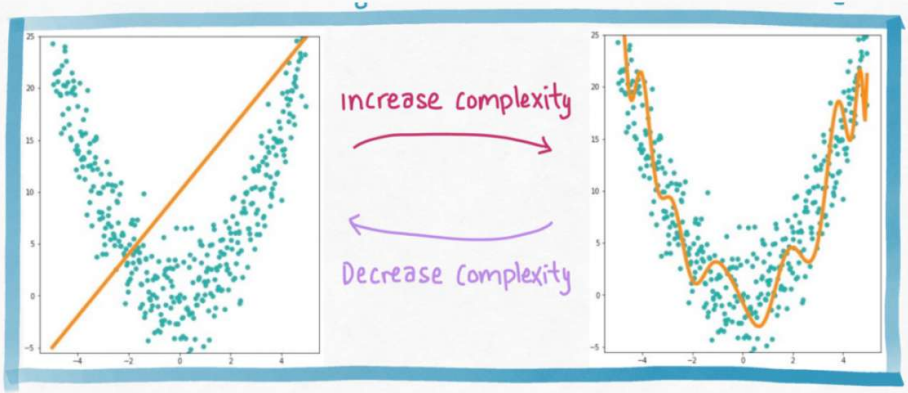
Bias vs Variance

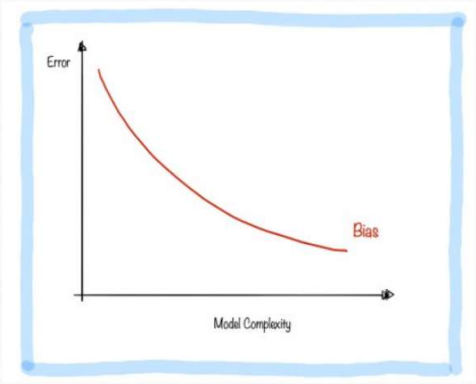
Left: 2000 best fit straight lines, each fitted on a different 20-points training set.

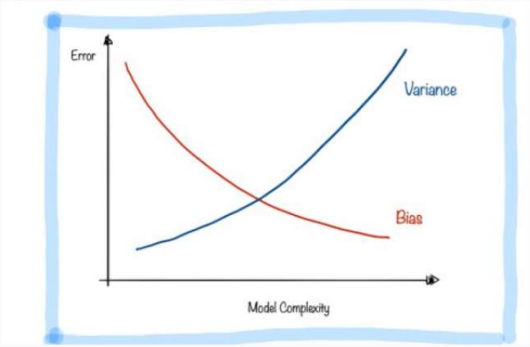
Right: Best-fit models using degree 10 polynomials.

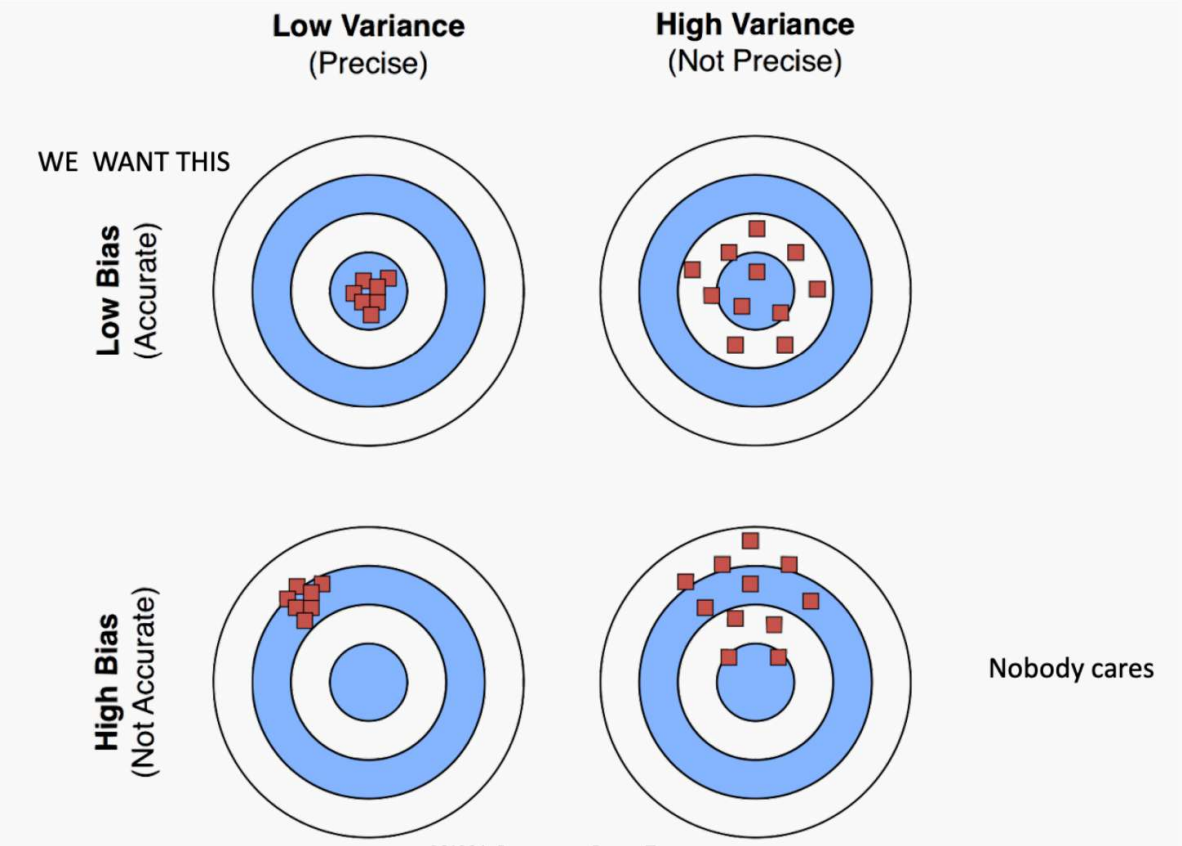


The Bias-Variance Trade Off: Bias








Low Variance
(Precise)
High Variance
(Not Precise)
WE WANT THIS
Low Bias
(Accurate)
High Bias
(Not Accurate)
Nobody cares

Overfitting

Overfitting occurs when a model corresponds too closely to the training set, and as a result, the model fails to fit additional data.

So far, we have seen that overfitting can happen when:

- Too many parameters
- Degree of the polynomial is too large
- Too many interaction terms

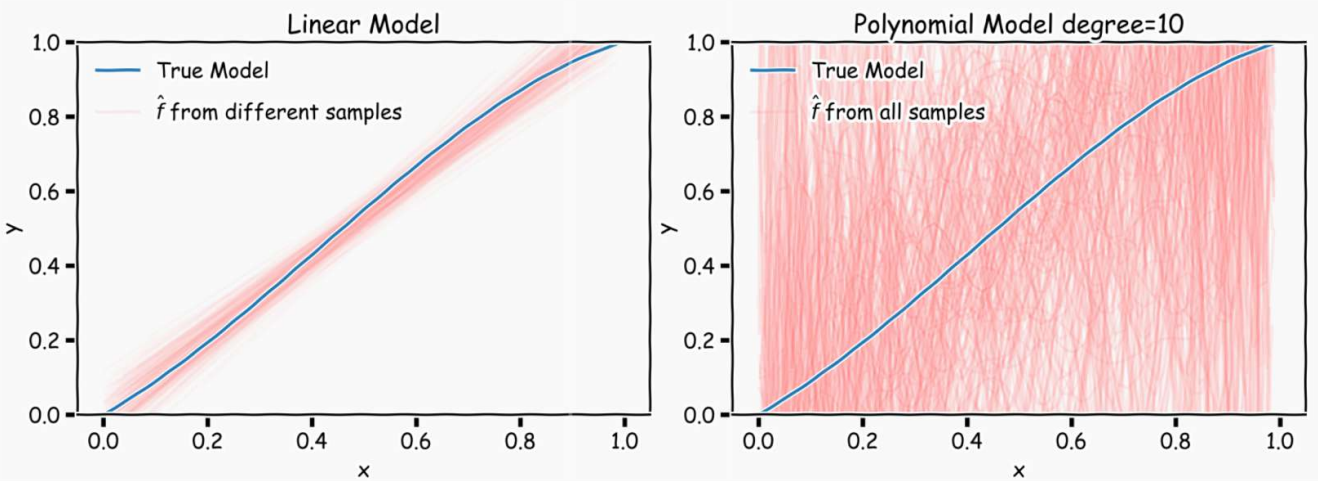
Next, we will see other evidence of overfitting, which will point to a way of avoiding overfitting: Ridge and Lasso regressions.

Ridge and Lasso

Bias vs Variance

Left: 2000 best fit straight lines, each fitted on a different 20-points training set.

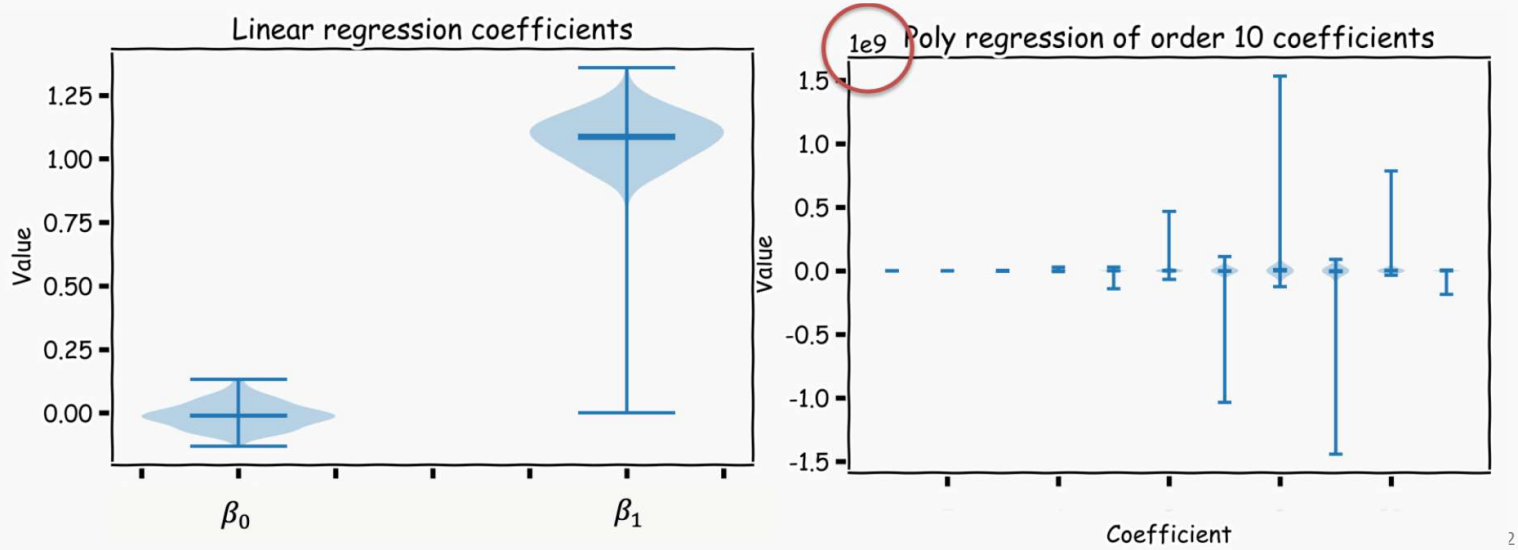
Right: Best-fit models using degree 10 polynomials.



Bias vs Variance

Left: Linear regression coefficients

Right: Poly regression of order 10 coefficients



Regularization: An Overview

The idea of regularization revolves around modifying the loss function L; in particular, we add a regularization term that penalizes some specified properties of the model parameters

$$L_{reg}(\beta) = L(\beta) + \lambda R(\beta)$$

where λ is a scalar that gives the weight (or importance) of the regularization term.

Fitting the model using the modified loss function L_{reg} would result in model parameters with desirable properties (specified by R).

LASSO Regression

Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.

Together our regularized loss function is:

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J}|\beta_j|.$$

Note that $\sum_{j=1}^{J}|\beta_j|$ is the $\boldsymbol{l_1}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J}|\beta_j| = \|\boldsymbol{\beta}\|_1$$

Ridge Regression

Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} \beta_j^2.$$

Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the $\boldsymbol{l_2}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

Choosing λ

In both ridge and LASSO regression, we see that the larger our choice of the **regularization parameter** λ, the more heavily we penalize large values in β,

- If λ is close to zero, we recover the MSE, i.e. ridge and LASSO regression is just ordinary regression.
- If λ is sufficiently large, the MSE term in the regularized loss function will be insignificant and the regularization term will force β_{ridge} and β_{LASSO} to be close to zero.

To avoid ad-hoc choices, we should select λ using validation or better cross-validation.

Regularization Parameter with a Validation Set

The solution of the Ridge/Lasso regression involves three steps:

- Select λ
- Find the minimum of the ridge/Lasso regression loss function (using the formula for ridge) and record the MSE **on the validation set**.
- Find the λ that gives the **smallest MSE on the validation set**.

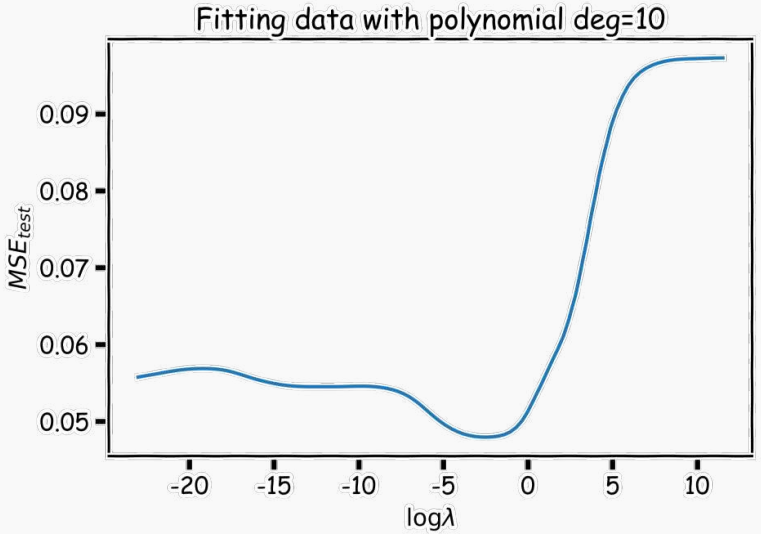
Ridge regularization with **validation** only: step by step

For ridge regression there exist an analytical solution for the coefficients:

$$\hat{\beta}_{Ridge}(\lambda) = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda I\right)^{-1} X^T Y$$

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots \lambda_{max}\}$:
    1. determine the β that minimizes the L_{ridge}, $\hat{\beta}_{Ridge}(\lambda) = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda I\right)^{-1} X^T Y$, using the train data.
    2. record $L_{MSE}(\lambda)$ using validation data.
3. select the λ that minimizes the loss on the validation data,
   $\lambda_{ridge} = \mathrm{argmin}_{\lambda}\, L_{MSE}(\lambda)$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with **validation** only: step by step



Lasso regularization with **validation** only: step by step

For Lasso regression there **not** an analytical solution for the coefficients so we use a **solver**.

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots \lambda_{max}\}$:
   A. determine the β that minimizes the L_{lasso}, $\hat{\beta}_{lasso}(\lambda)$, using the train data. **This is done using a solver.**
   B. record $L_{MSE}(\lambda)$ using validation data
3. select the λ that minimizes the loss on the validation data,
   $\lambda_{lasso} = \text{argmin}_{\lambda}\, L_{MSE}(\lambda)$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{lasso}(\lambda_{lasso})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{lasso}(\lambda_{lasso})$

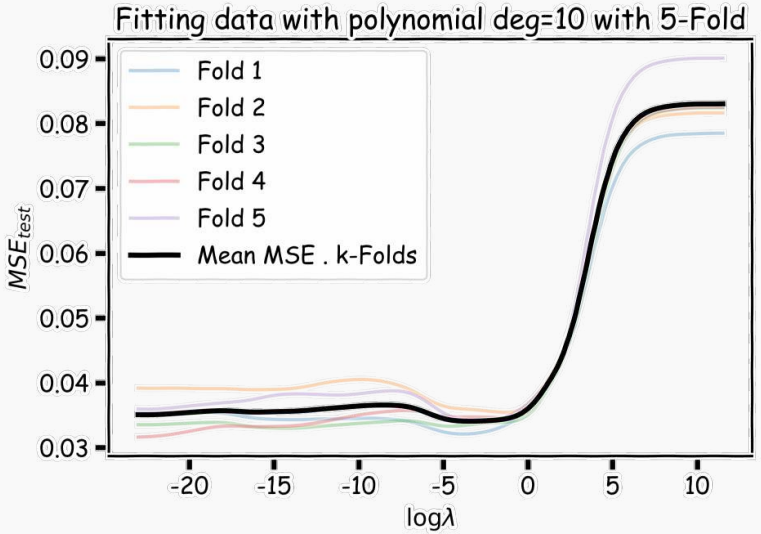
Ridge regularization with **CV**: step by step

	λ_1	λ_2	...	λ_n
k_1	L_{11}	L_{12}	..	...
k_2	L_{21}	...	..	...
...	..	...	..	...
k_n	...	...	...	...
E[]	$\bar{L}_1$	$\bar{L}_2$	...	$\bar{L}_n$

1. remove $\{X,Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X,Y\}_{train}^{-k}, \{X,Y\}_{val}^{k}\}$
3. for k in $\{1,\dots,K\}$
    1. for λ in $\{\lambda_0, \dots, \lambda_n\}$:
        A. determine the β that minimizes the L_{ridge}, $\hat{\beta}_{ridge}(\lambda,k) = \left(\mathrm{X^T X} + \lambda I\right)^{-1} X^T Y$, using the train data of the fold, $\{X,Y\}_{train}^{-k}$.
        B. record $L_{MSE}(\lambda,k)$ using the validation data of the fold $\{X,Y\}_{val}^{k}$

        At this point we have a 2-D matrix, rows are for different k, and columns are for different λ values.
4. Average the $L_{MSE}(\lambda,k)$ for each λ, $\bar{L}_{MSE}(\lambda)$.
5. Find the λ that minimizes the $\bar{L}_{MSE}(\lambda)$, resulting to λ_{ridge}.
6. Refit the model using the full training data, $\{\{X,Y\}_{train}, \{X,Y\}_{val}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
7. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with **validation** only: step by step



Ridge and Lasso Comparision

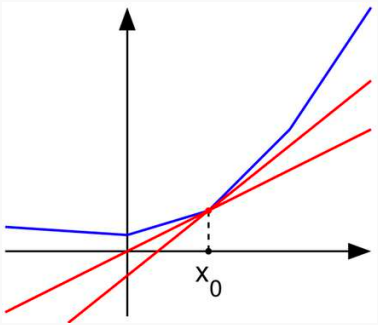
Ridge, LASSO - Computational complexity

Solution to ridge regression:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

The solution to the LASSO regression:

LASSO has no conventional analytical solution, as the L1 norm has no derivative at 0. We can, however, use the concept of subdifferential or subgradient to find a manageable expression.

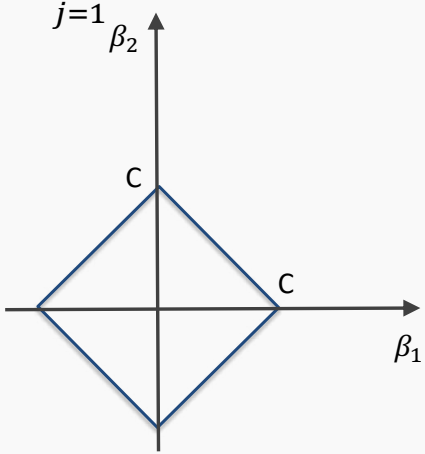


The Geometry of Regularization (LASSO)

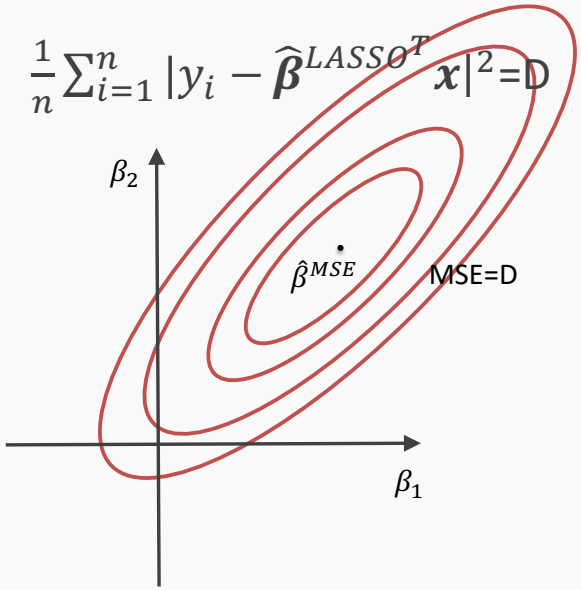
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}\Big|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \operatorname{argmin} L_{LASSO}(\boldsymbol{\beta})$$

$$\lambda \sum_{j=1}^{J} |\hat{\beta}_j^{LASSO}| = C$$



$$\frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{\boldsymbol{\beta}}^{LASSO^T} \boldsymbol{x}|^2 = D$$

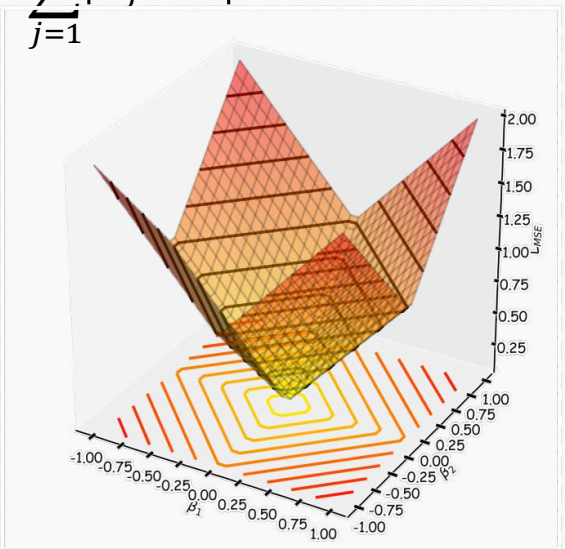


The Geometry of Regularization (LASSO)

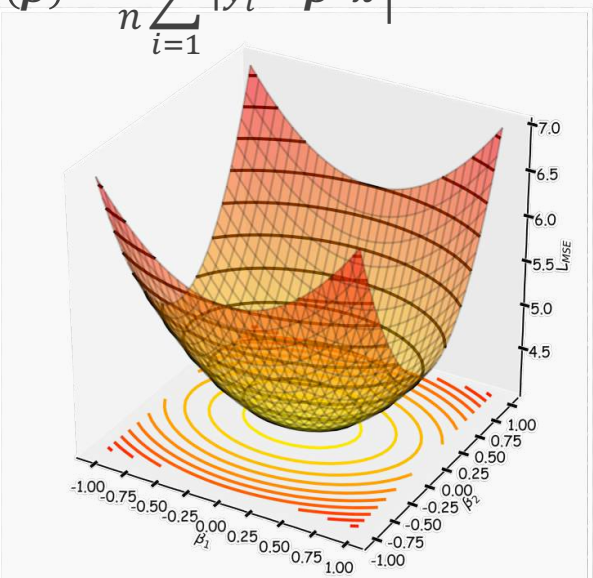
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}\Big|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \operatorname{argmin} L_{LASSO}(\boldsymbol{\beta})$$

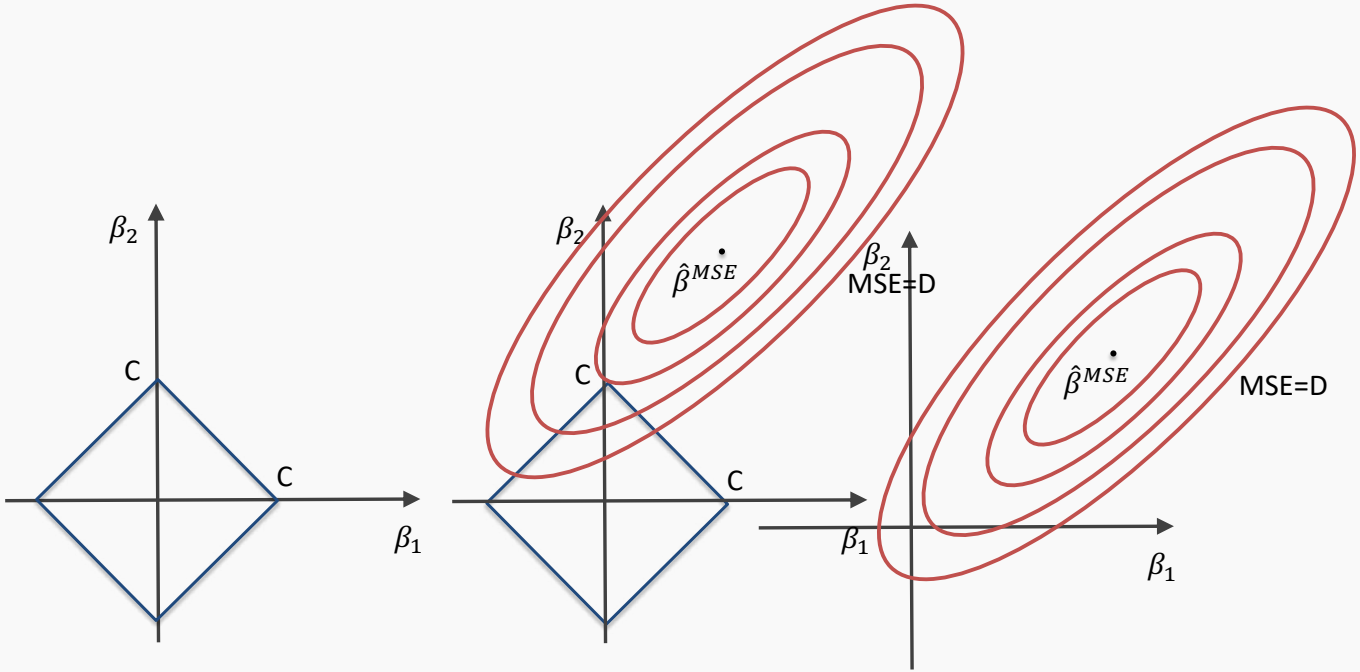
$$L_1 = \lambda \sum_{j=1}^{J} \Big| \hat{\beta}_j^{LASSO} \Big|$$



$$L_{MSE}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}\Big|^2$$



The Geometry of Regularization (LASSO)

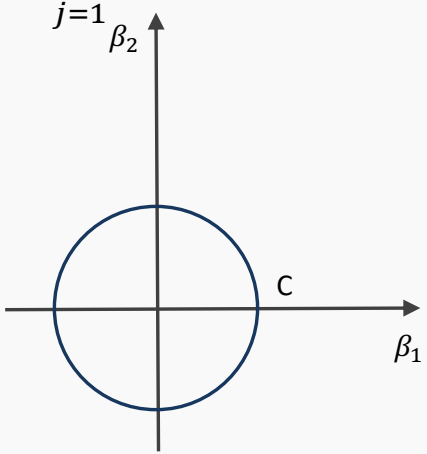


The Geometry of Regularization (Ridge)

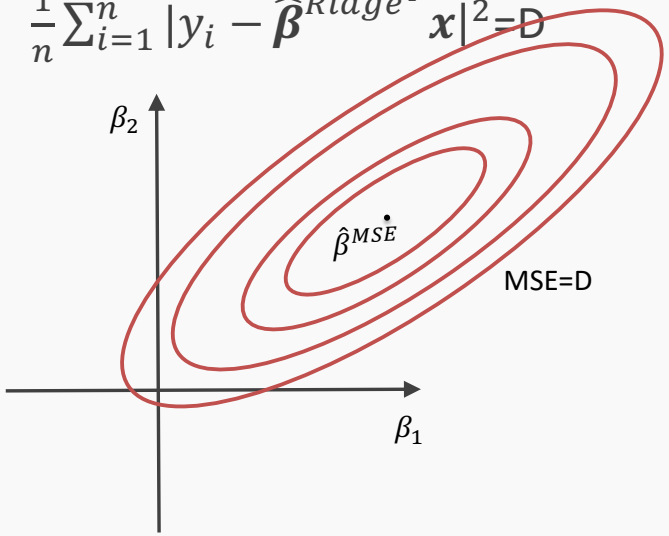
$$L_{Ridge}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}|^2 + \lambda \sum_{j=1}^{J} (\beta_j)^2$$

$$\widehat{\boldsymbol{\beta}}^{Ridge} = \operatorname{argmin} L_{Ridge}(\boldsymbol{\beta})$$

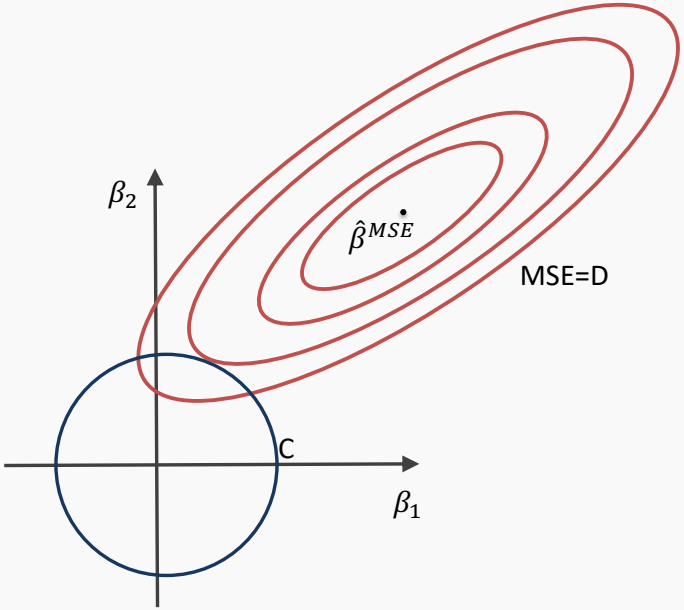
$$\lambda \sum_{j=1}^{J} \left|\hat{\beta}_j^{Ridge}\right|^2 = C$$



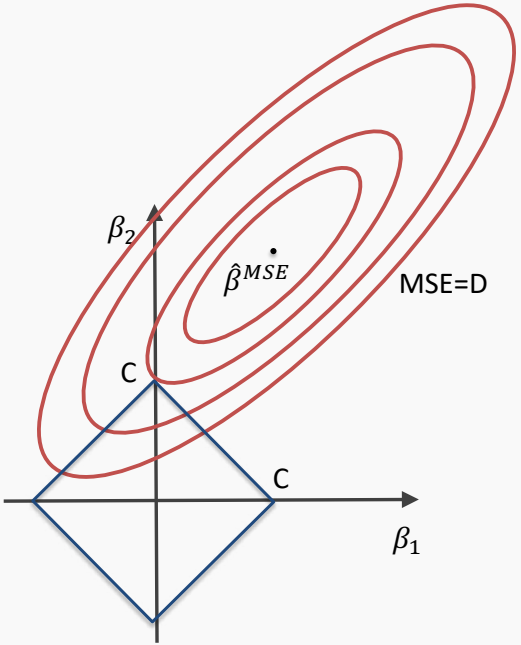
$$\frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{\boldsymbol{\beta}}^{Ridge^T} \boldsymbol{x}|^2 = D$$

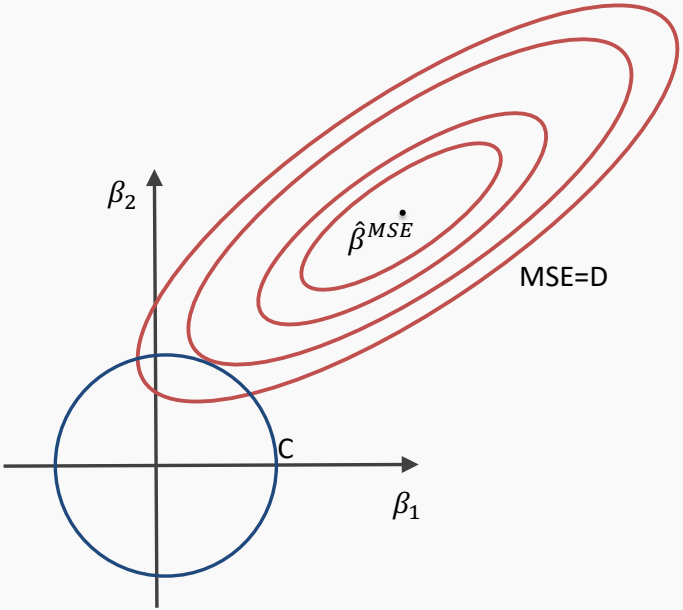


The Geometry of Regularization (Ridge)

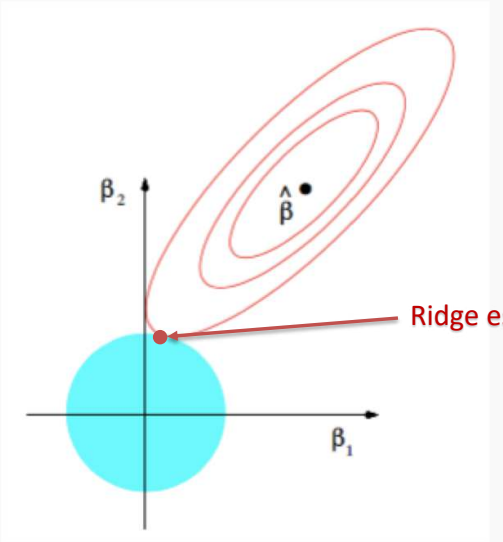


The Geometry of Regularization

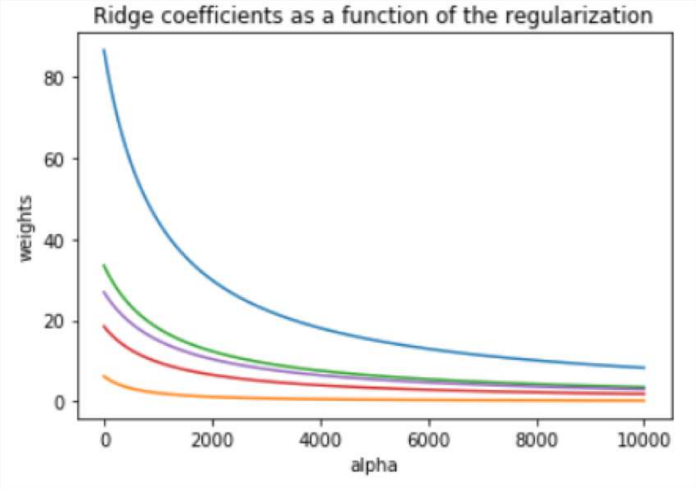




Ridge visualized

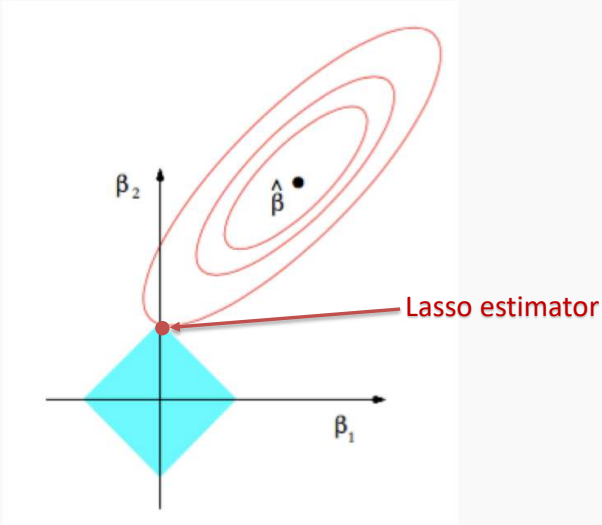


The ridge estimator is where the constraint and the loss intersect.

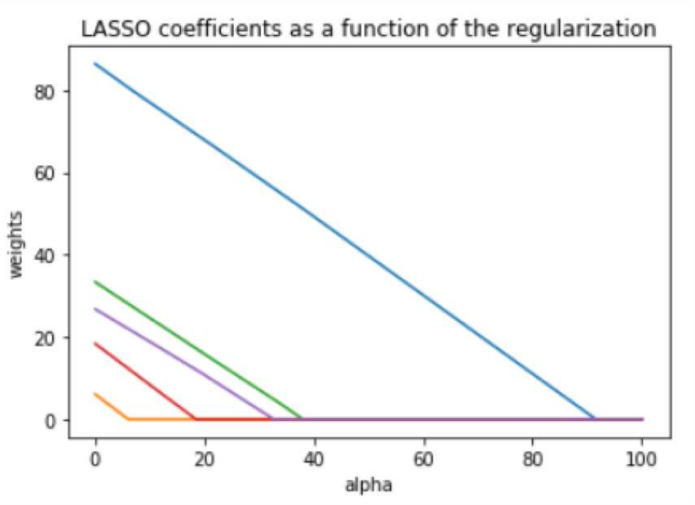


The values of the coefficients decrease as lambda increases, but they are not nullified.

LASSO visualized



The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axis.



The values of the coefficients decrease as lambda increases, and are nullified fast.

Variable Selection as Regularization

Question: What are the pros and cons of the two approaches?

Since LASSO regression tends to produce zero estimates for a number of model parameters - we say that LASSO solutions are **sparse** - we consider LASSO to be a variable selection method.

Ridge is faster to compute but many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) and therefore easier to interpret.