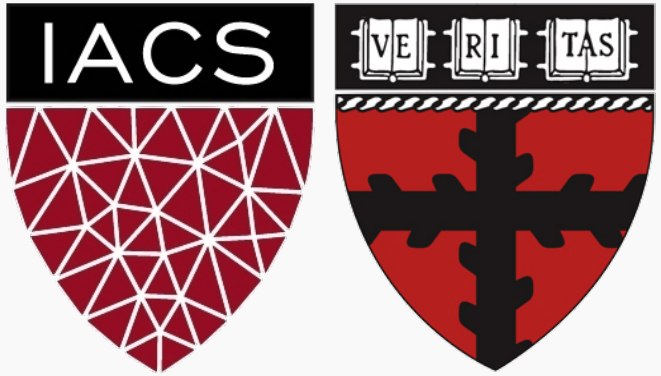
Lecture 7: Model Selection and Regularization

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner



ANNOUNCEMENTS

Study break next week (TBD):

- Pizza and fun

Homework:

- HW1 Grades will be released sometime today. Median = 4.8, STD= 0.60
- HW3: 2 weeks and thus longer.
- Read the submission instructions – **please**.
- Use the pre-made groups

Normally, you need only submit the **.ipynb file** with one caveat:

If your notebook references other files such as images which are necessary for interpreting your work, they must be uploaded as well.

You may also choose to "attach" the images to the notebook itself. See the section on markdown attachments under "other additions" here for details.

File Naming Convention

Name your file as follows: cs109a_hw#_?09_submit.ipynb (replace the # with the number of the assignment and replace ? with 1 or 2 depending if you are submitting the 109 or 209 hw). **Do not include your name** in the paper or the filename as we are doing anonymous grading.

Making a HW Group in Canvas

DO NOT make new groups. Look for an empty group among the **pre-made** groups for each assignment and join it. Follow instructions below.

1. Click "People" in the navbar
2. Click the "Groups" tab
3. Type in the search box: "HW#" (replace # with the homework number, e.g. HW3) then wait a long time for the groups to load
4. Choose an empty group and have both members in your pair click "Join". Again, this must be done **before** you submit and **before** the homework deadline. If you haven't joined a group before the homework deadline, we will assume you are working alone.
5. Groups **do not** persist across assignments. You need to make a group for every assignment.

Warning: If you are submitting in a pair: both members must join a Canvas group BEFORE they submit and BEFORE the homework deadline, whichever is earlier. If your group is not in place when you submit or if you submit after the assignment deadline, then you are submitting alone. This is a Canvas limitation.

Making a HW Group in Canvas Video

ANNOUNCEMENTS

Projects:

1. Acquiring, organizing, and working with **real data.**
2. **Collaborating** with your peers.
3. **Integrating** statistics and machine learning methods.
4. Working with an **open ended** problems. Find optimizable second-best approach for problems that can NOT be solved.
5. **Communicating** your work to others.
6. Engage fully with the Data Science process—in all its **non-linearity**—in a real world project situation.

Polynomial Regression

The simplest non-linear model we can consider, for a response Y and a predictor X, is a polynomial model of degree M,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_M x^M + \epsilon.$$

Just as in the case of linear regression with cross terms, polynomial regression is a special case of linear regression - we treat each x^m as a separate predictor. Thus, we can write

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_1^M \\ 1 & x_2^1 & \ldots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^M \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

Polynomial Regression

Multi-Regression

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,J} \\ 1 & x_{2,1} & \dots & x_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,J} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_J \end{pmatrix},$$

Poly-Regression

If $TV is one variable, then ($TV)**n is another variable.

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^M \\ 1 & x_2^1 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^M \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

Lecture Outline

Overfitting

Model Selection

Cross Validation

Bias vs Variance

Regularization: LASSO and Ridge

Regularization Methods: A Comparison

Lecture Outline

Overfitting

Model Selection

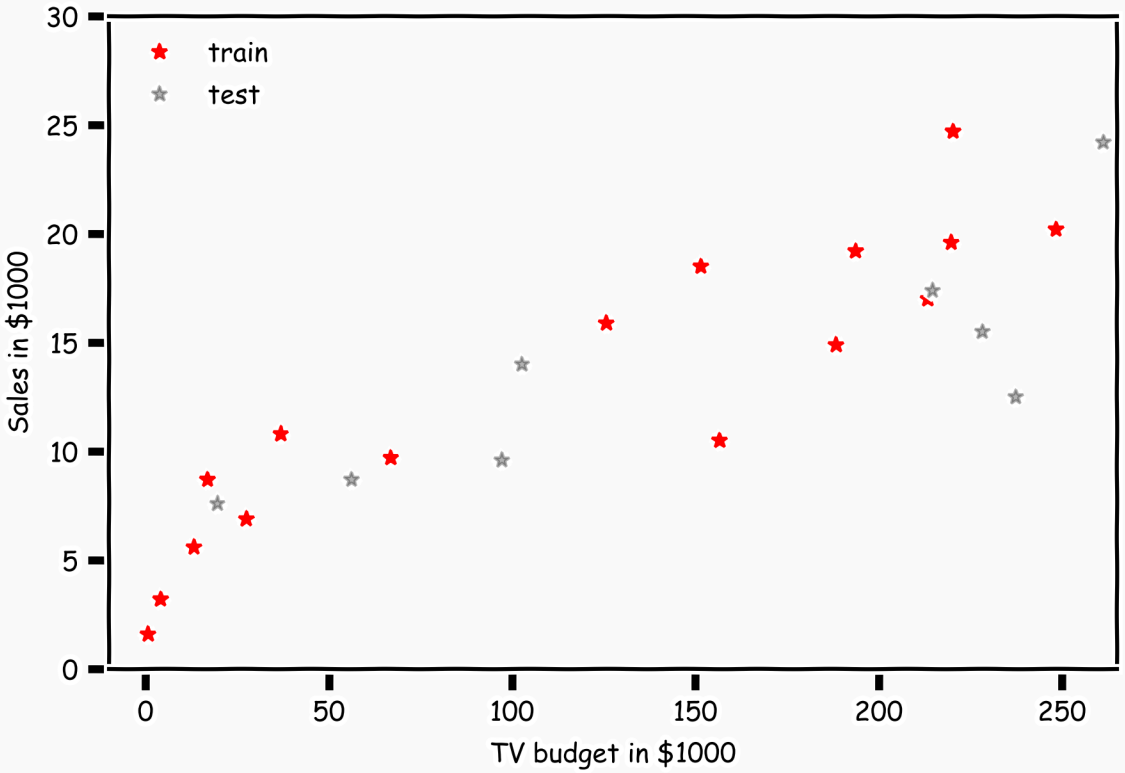
Cross Validation

Bias vs Variance

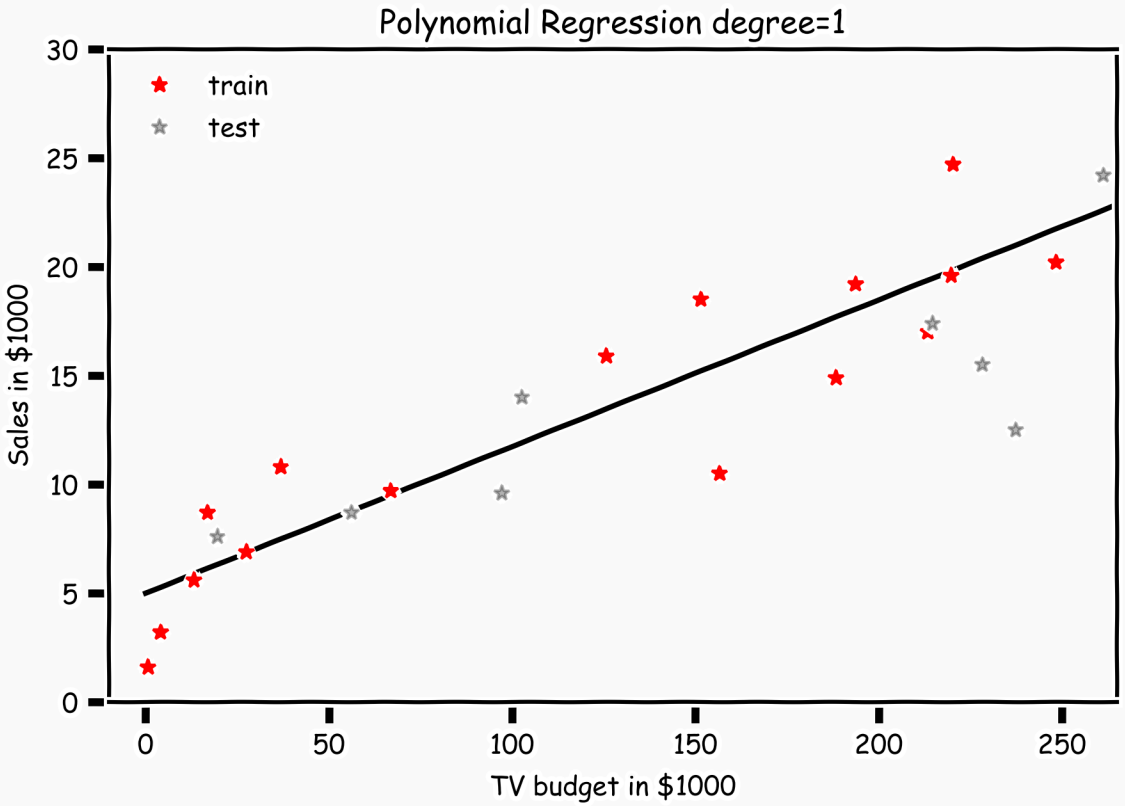
Regularization: LASSO and Ridge

Regularization Methods: A Comparison

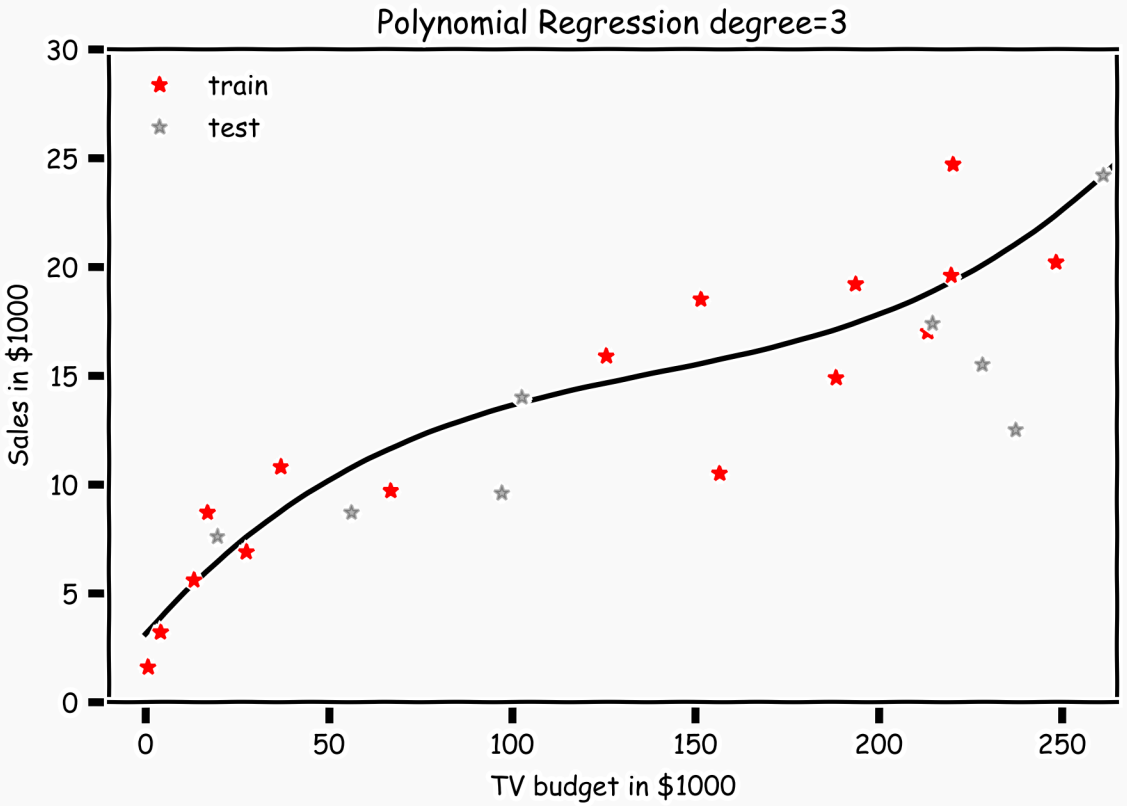
Overfitting



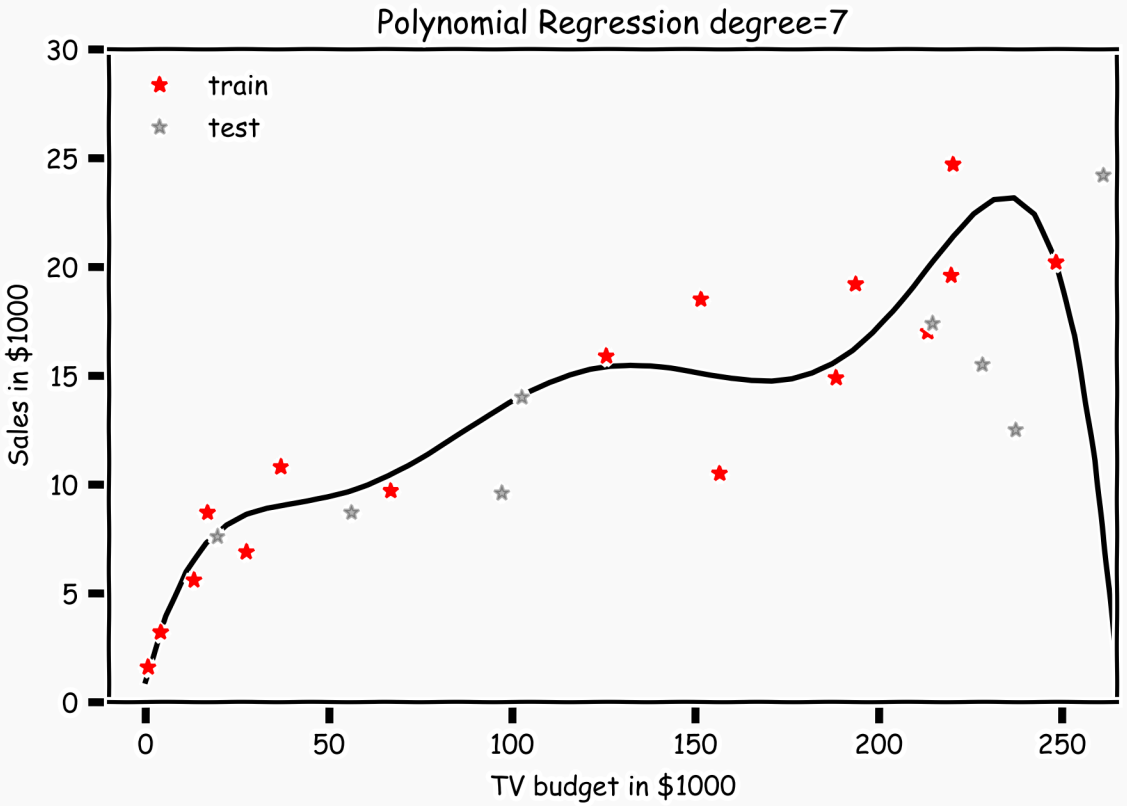
Overfitting



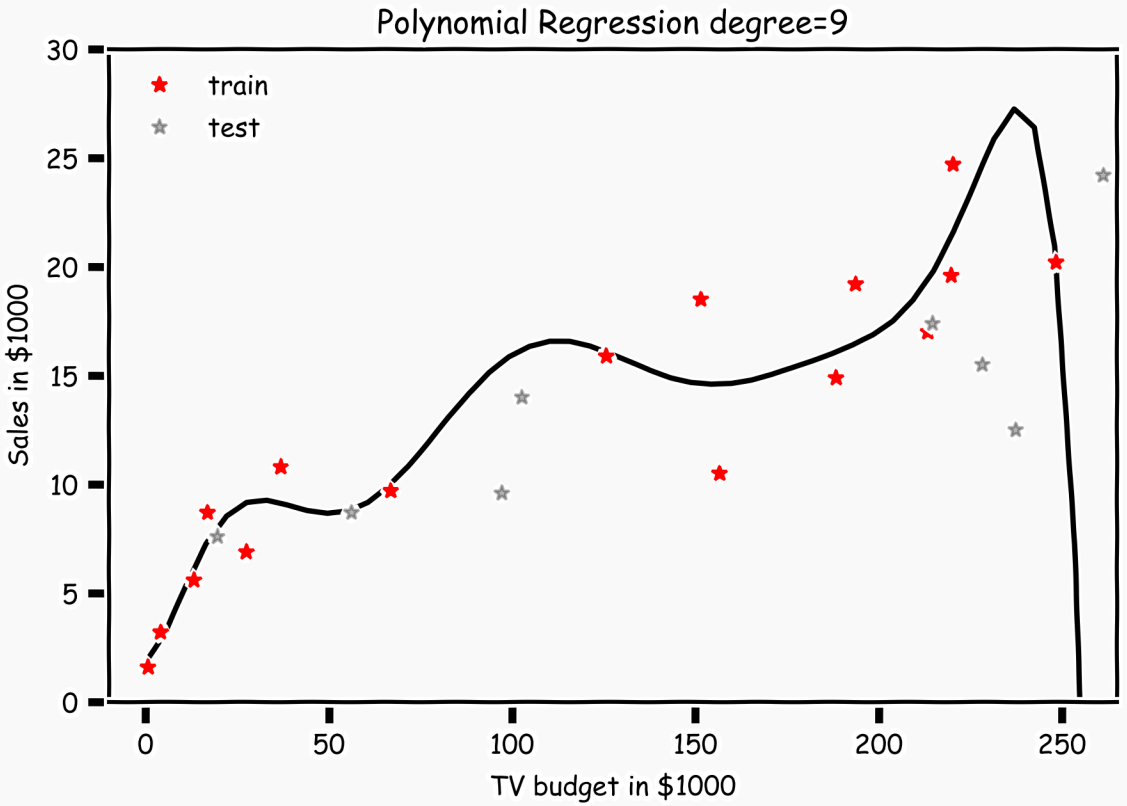
Overfitting



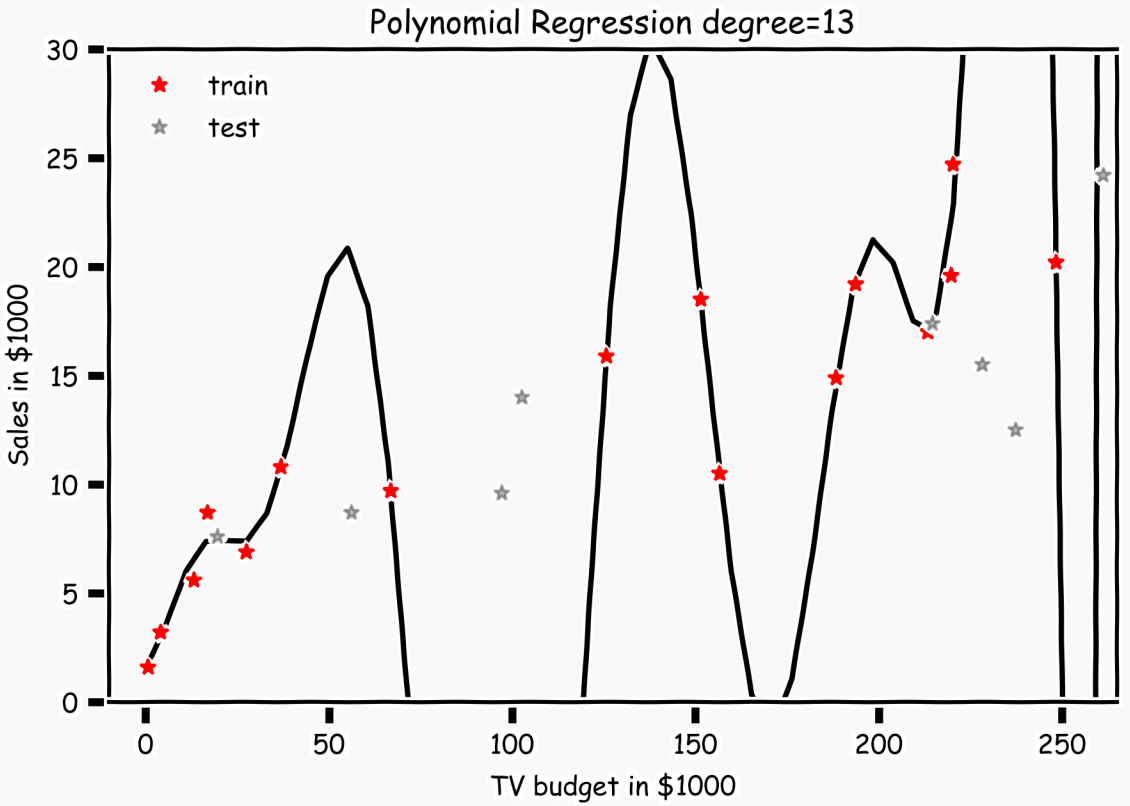
Overfitting



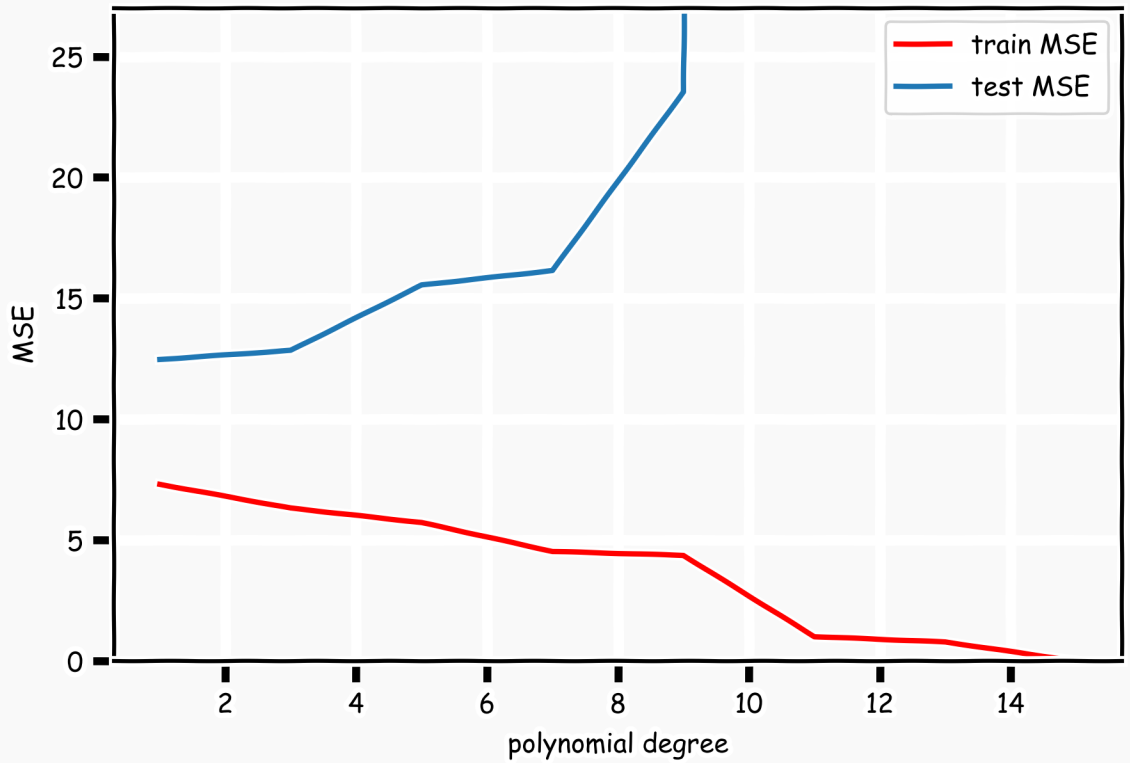
Overfitting



Overfitting



Validation


train MSE
test MSE
MSE
0
5
10
15
20
25
2
4
6
8
10
12
14
polynomial degree

Train-Validation-Test

Question:

How would you report the performance of the model?

R2_test = 0.52

R2_train(degree=1) = 0.83

Train: fit the model
Validation: selection the model
Test: report the result

Overfitting: performance on the train and on the validation is not the same.

Lecture Outline

Overfitting

Model Selection

Cross Validation

Bias vs Variance

Regularization: LASSO and Ridge

Regularization Methods: A Comparison

Model Selection

Model selection is the application of a principled method to determine the complexity of the model, e.g. choosing a subset of predictors, choosing the degree of the polynomial model etc.

A strong motivation for performing model selection is to avoid overfitting, which we saw can happen when:

- there are too many predictors:
  - the feature space has high dimensionality
  - the polynomial degree is too high
  - too many cross terms are considered
- the coefficients values are too **extreme (we have not seen this yet)**

Model Selection

Question:

How many different models when considering J predictors?

Model Selection

Example: 3 predictors (X_1, X_2, X_3)

- Models with 0 predictor:

  M0:

- Models with 1 predictor:

  M1: X_1

  M2: X_2

  M3: X_3

- Models with 2 predictors:

  M4: $\{X_1, X_2\}$

  M5: $\{X_2, X_3\}$

  M6: $\{X_3, X_1\}$

- Models with 3 predictors:

  M7: $\{X_1, X_2, X_3\}$

2^J Models

Stepwise Variable Selection and Cross Validation

Selecting optimal subsets of predictors (including choosing the degree of polynomial models) through:

- stepwise variable selection - iteratively building an optimal subset of predictors by optimizing a fixed model evaluation metric each time,
- validation - selecting an optimal model by evaluating each model on validation set.

We will also address the issue of discouraging extreme values in model parameters later.

Stepwise Variable Selection: Forward method

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

1. Start with the empty set P0, construct the null modelM0.

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/BIS on training set)

Stepwise Variable Selection Computational Complexity

How many models did we evaluate?

- 1st step, ***J* Models**
- 2nd step, *J-1* **Models** (add 1 predictor out of *J-1* possible)
- 3rd step, *J-2* **Models** (add 1 predictor out of *J-2* possible)
- …

$$O(J^2) \ll 2^J \text{ for large } J$$

Lecture Outline

Overfitting

Model Selection

Cross Validation

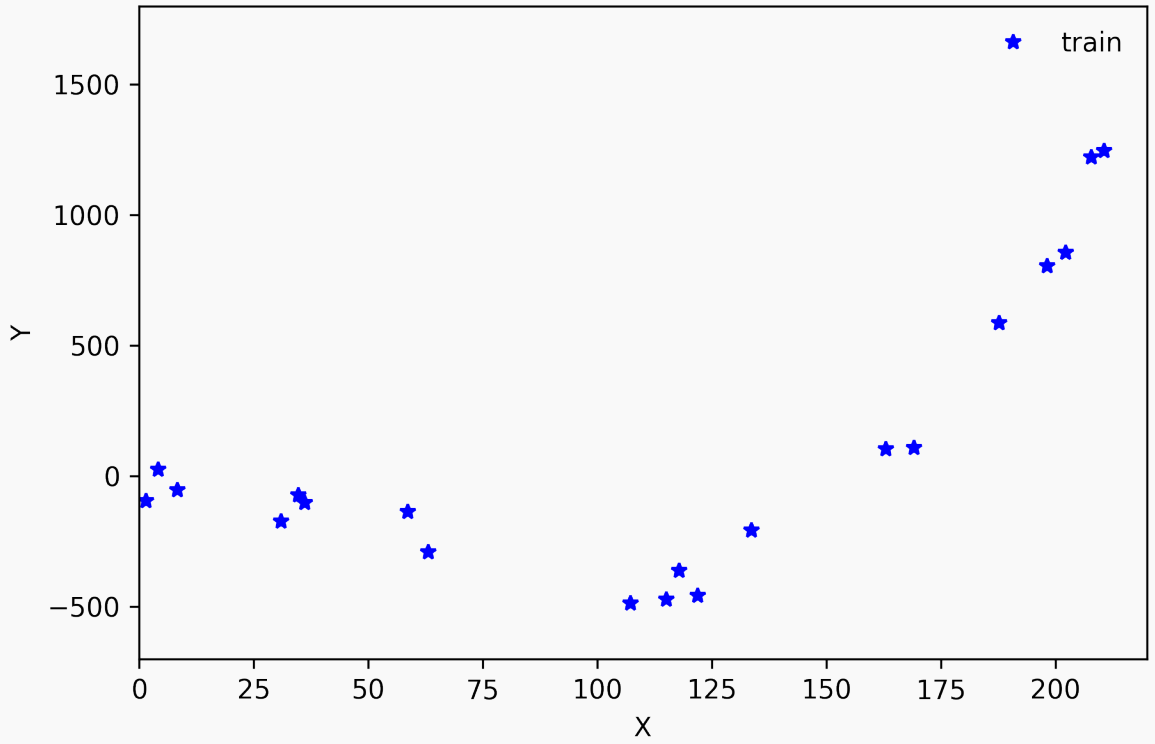
Bias vs Variance

Regularization: LASSO and Ridge

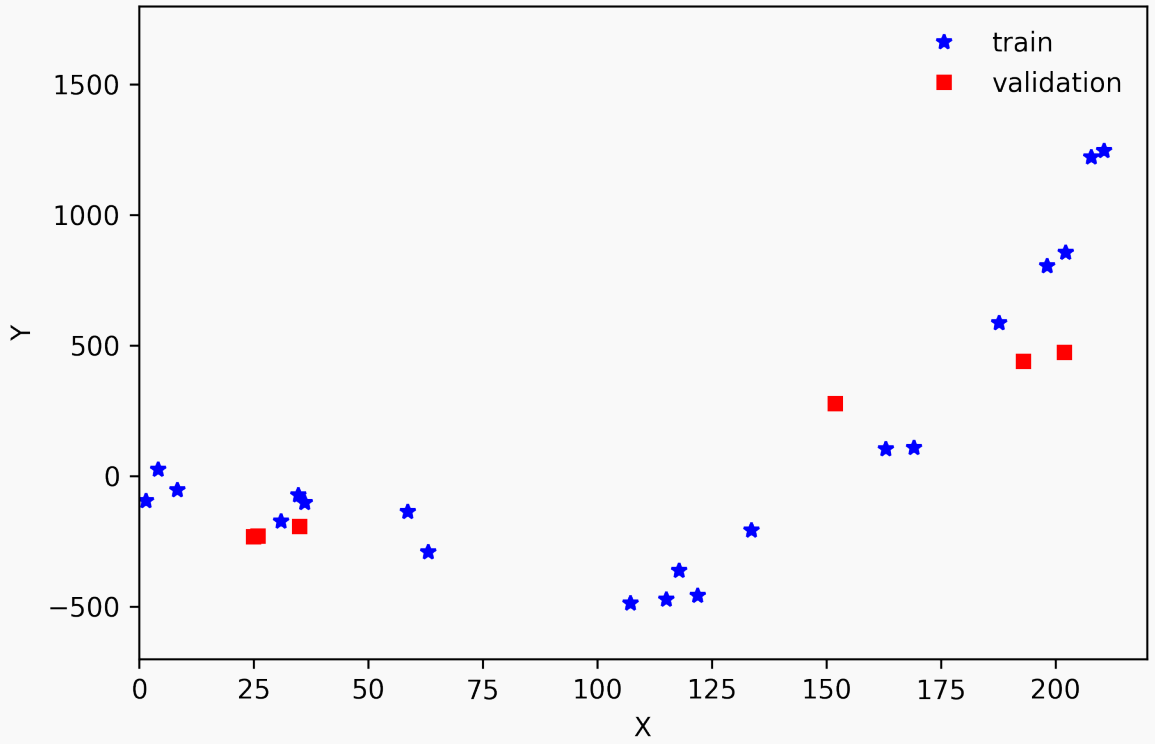
Regularization Methods: A Comparison

Cross Validation

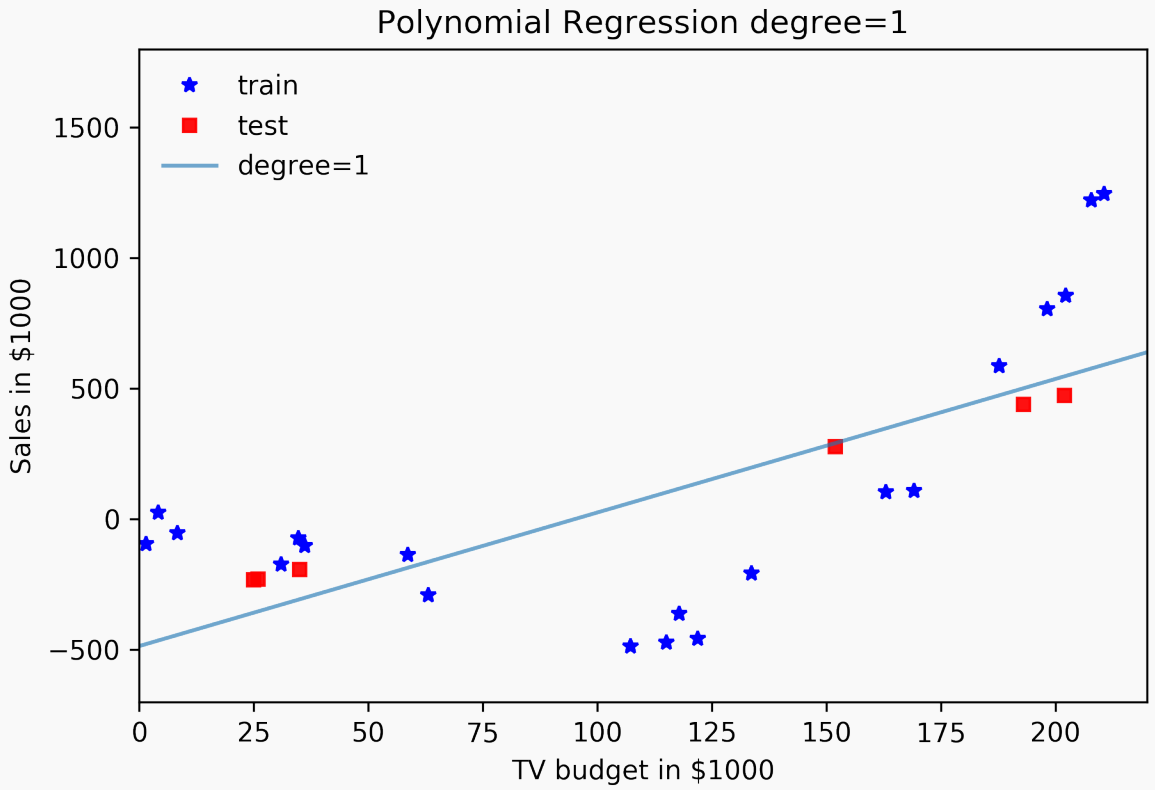
Cross Validation



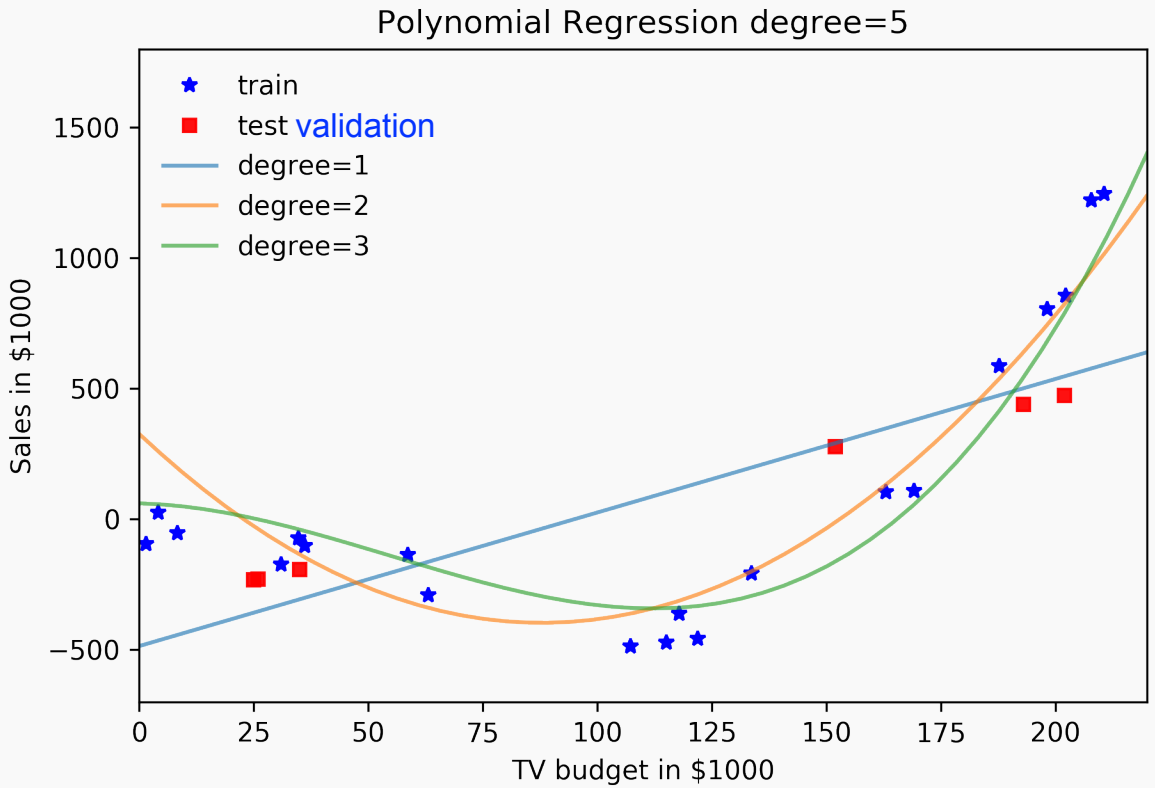
Cross Validation



Cross Validation



Cross Validation



The chance of getting strange validation sets increases at the number of the points in the validation set decreases.

Validation


train MSE
validation MSE
MSE
25
20
15
10
5
0
2.5
5.0
7.5
10.0
12.5
15.0
polynomial degree

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**.

One solution to the problems raised by using a single validation set is to evaluate each model on **multiple** validation sets and average the validation performance.

One can randomly split the training set into training and validation multiple times **but** randomly creating these sets can create the scenario where important features of the data never appear in our random draws.

Cross Validation

train-test
shuffle the train set
split the train set into different folds

for i in model[0:n]:
    use each of the fold sequentially for validation set
      - use model i to fit the train
      - use model i to test the validation
      - sum all the validation scores and take the mean
    the mean is the score for the model i.

find the best model
get the validation fold back to the train folds
fit the best model to the whole train set, get the score

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: $X_{-i} = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\}$

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

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.

Predictor Selection: Cross Validation

Question: What is the right ratio of train/validate/test, how do I choose K?

20%

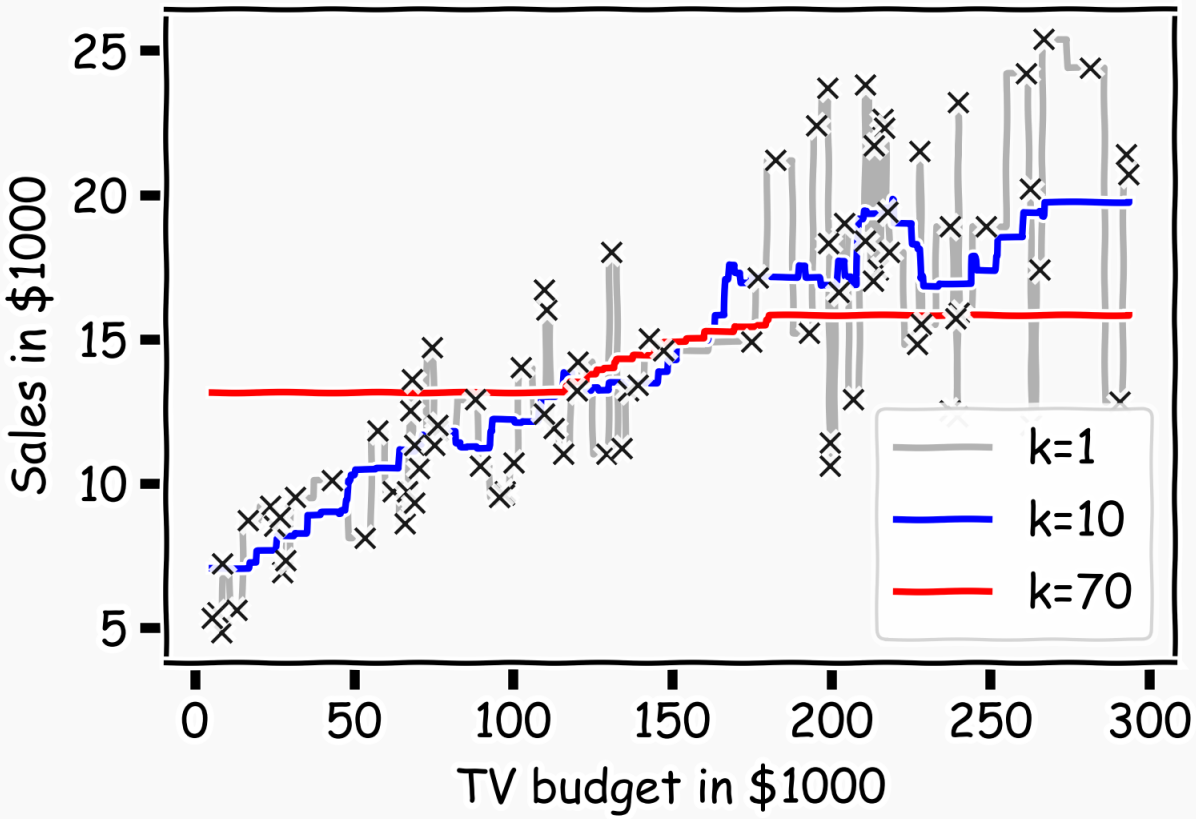
Question: What is the difference in multiple predictors and polynomial regression in model selection?

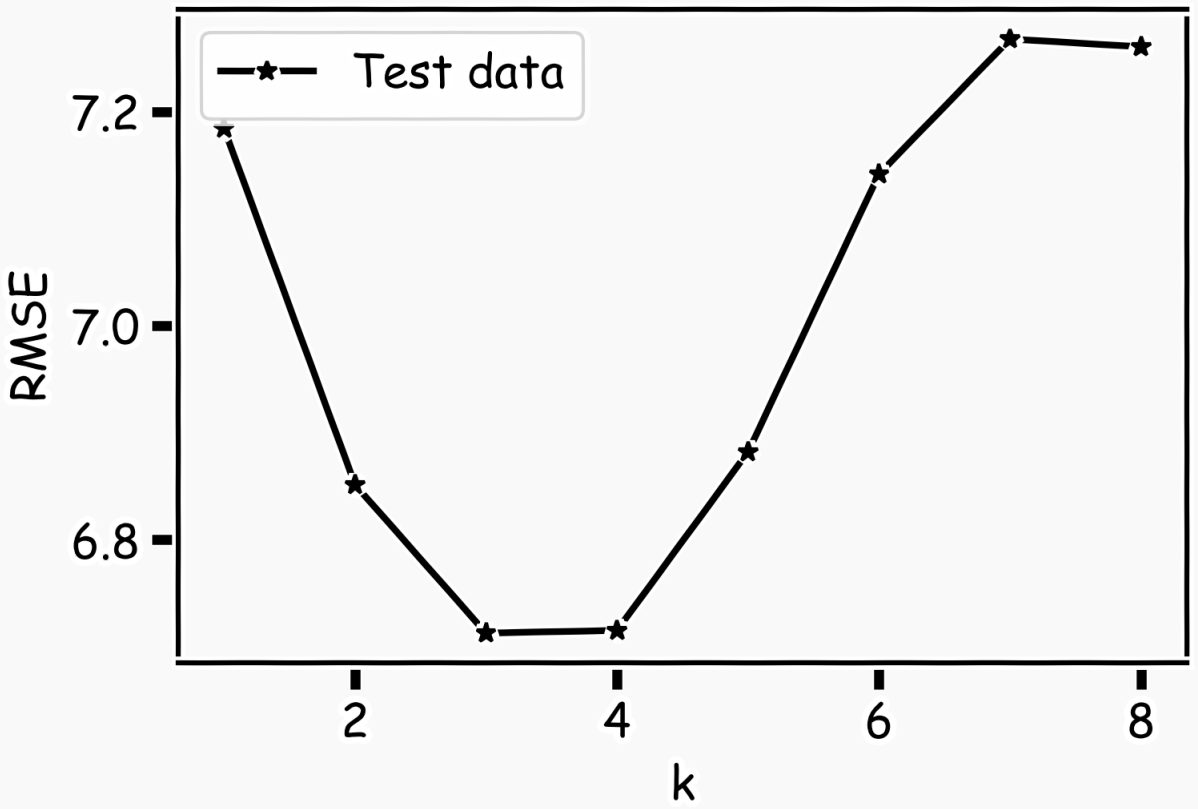
for multiple: we can choose the predictors as we wish
for polynomial: the power progresses one-by-one

We can frame the problem of degree selection for polynomial models as a predictor selection problem:

which of the predictors $\{x, x^2, \ldots, x^m\}$, should we select for modeling?

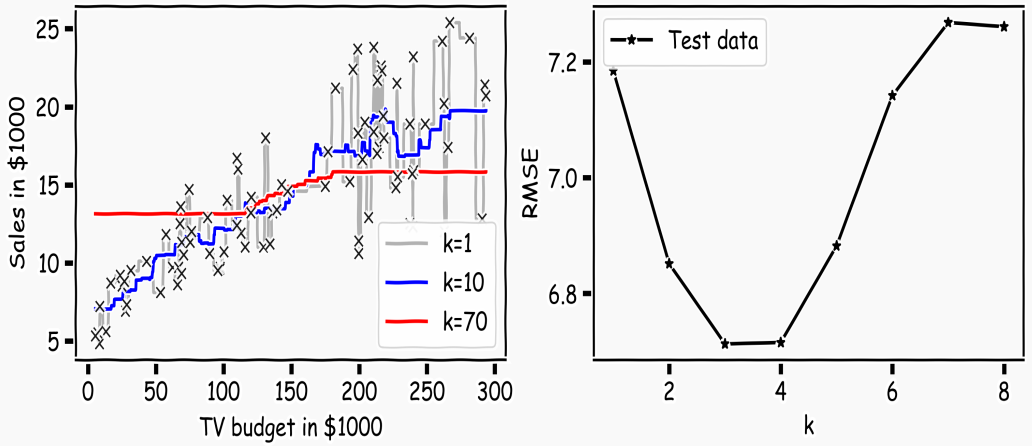
kNN Revisited





kNN Revisited

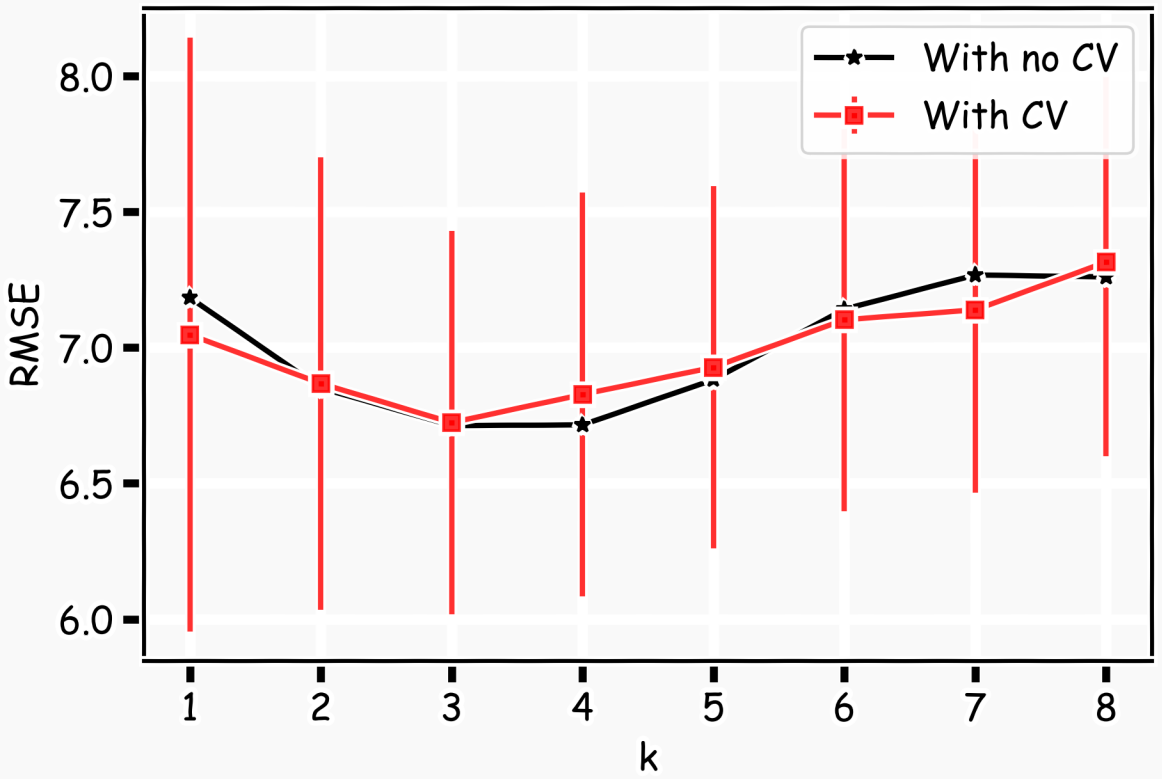
Recall our first simple, intuitive, non-parametric model for regression – the kNN model. We saw that it is vitally important to select an appropriate *k* for the data.



If the *k* is too small then the model is very sensitive to noise (since a new prediction is based on very few observed neighbors), and if the *k* is too large, the model tends towards making constant predictions.

A principled way to choose *k* is **through K-fold cross validation.**

K-fold with k=100



Lecture Outline

Overfitting

Model Selection

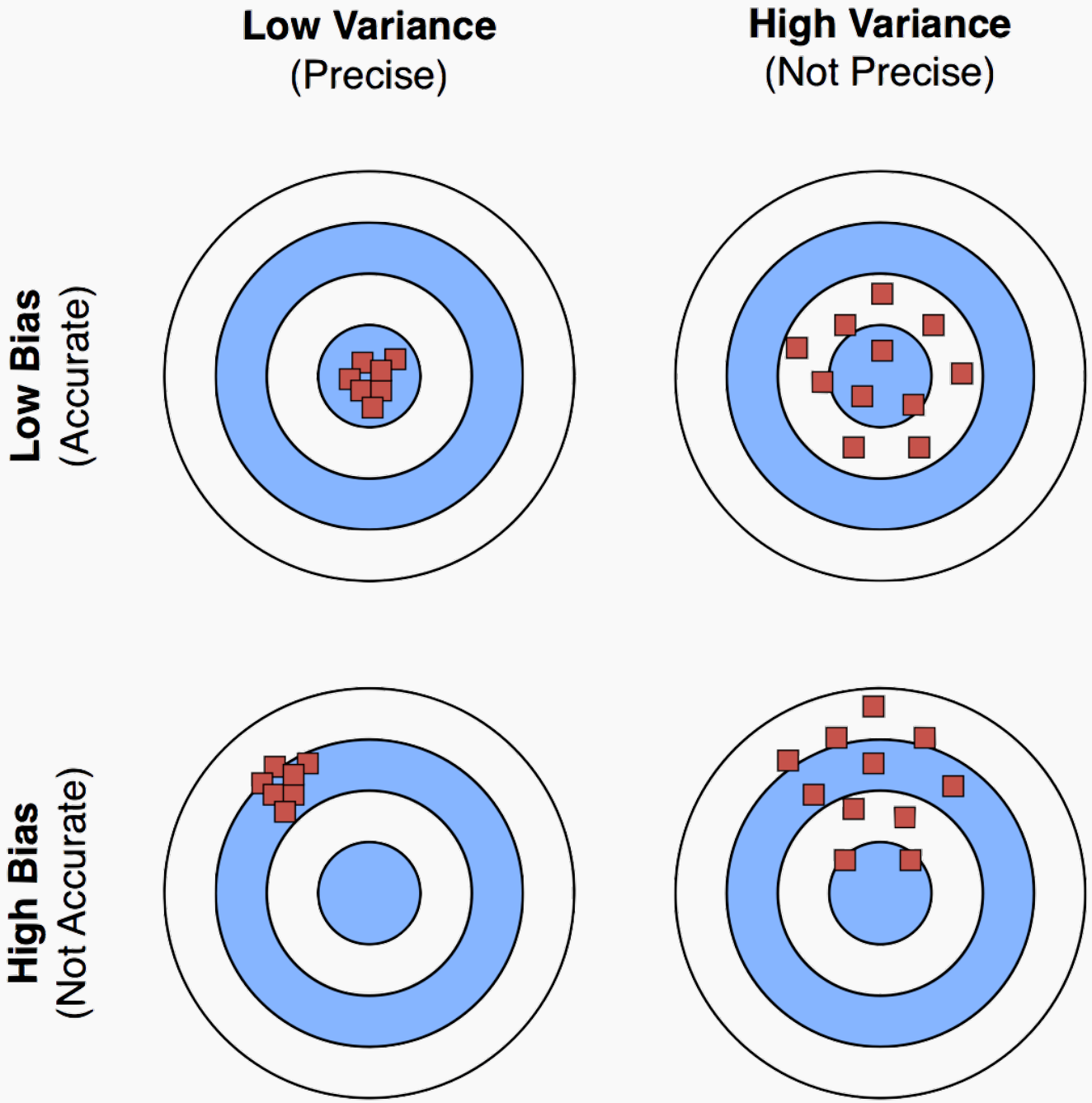
Cross Validation

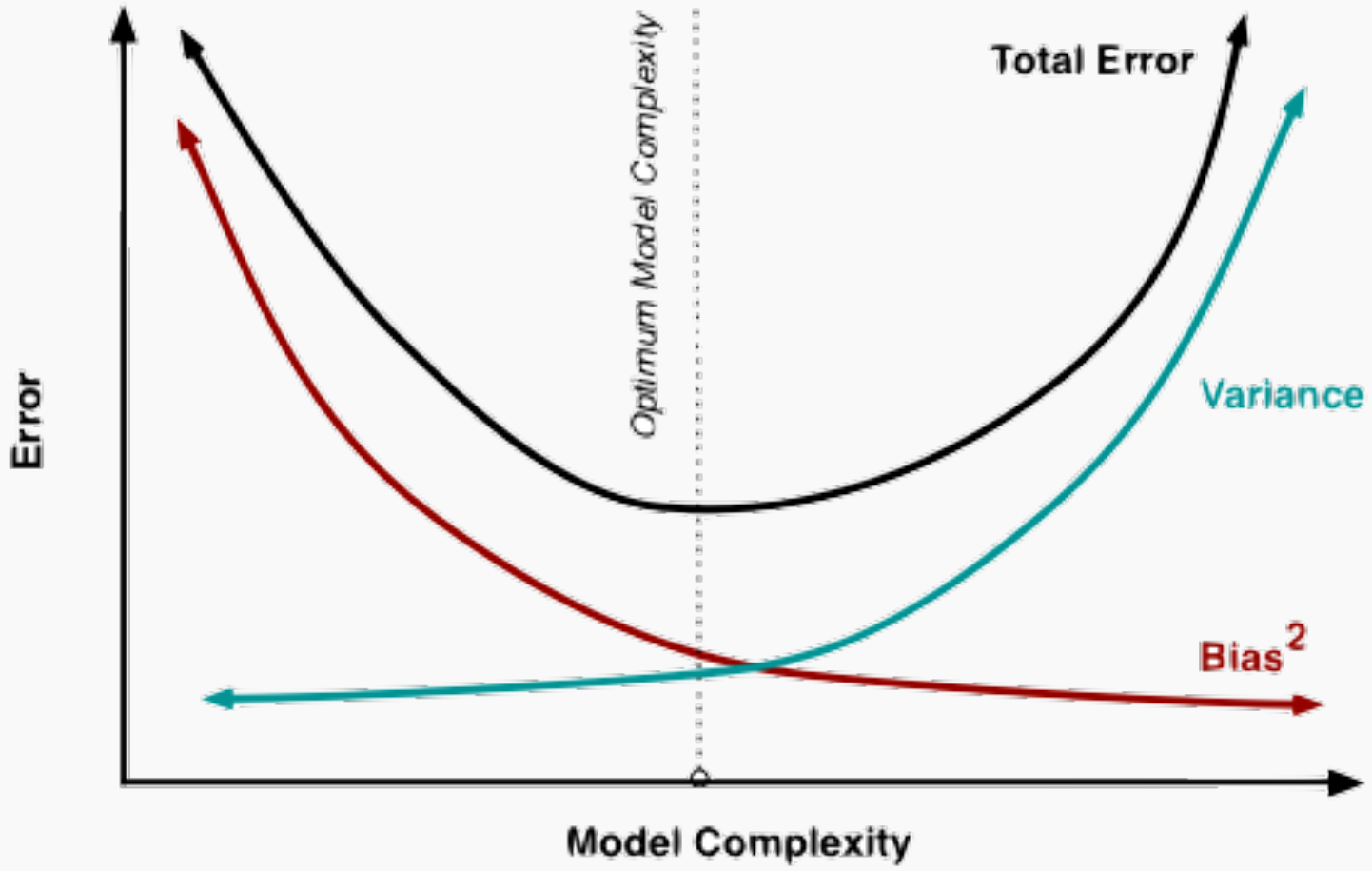
Bias vs Variance

Regularization: LASSO and Ridge

Regularization Methods: A Comparison

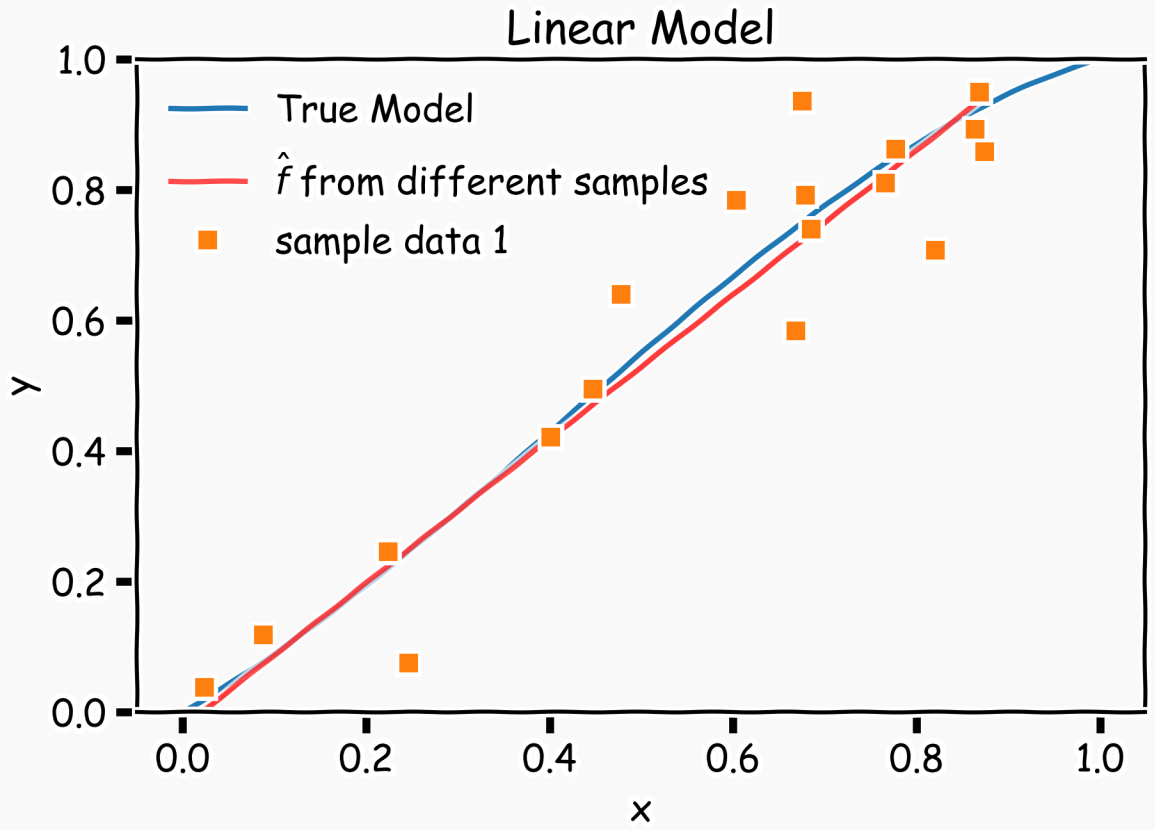
Bias vs Variance


Low Variance
(Precise)
High Variance
(Not Precise)
Low Bias
(Accurate)
High Bias
(Not Accurate)

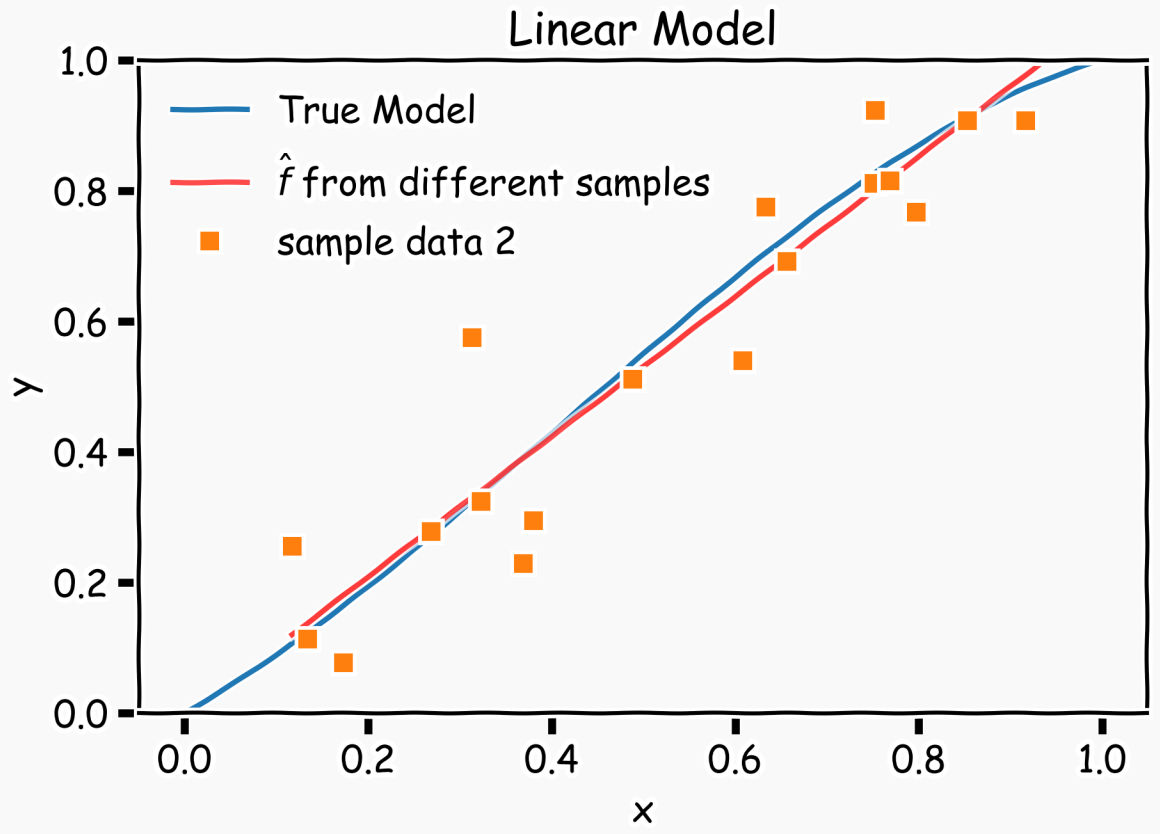

Error
Optimum Model Complexity
Total Error
Variance
Bias^2
Model Complexity

Bias vs Variance

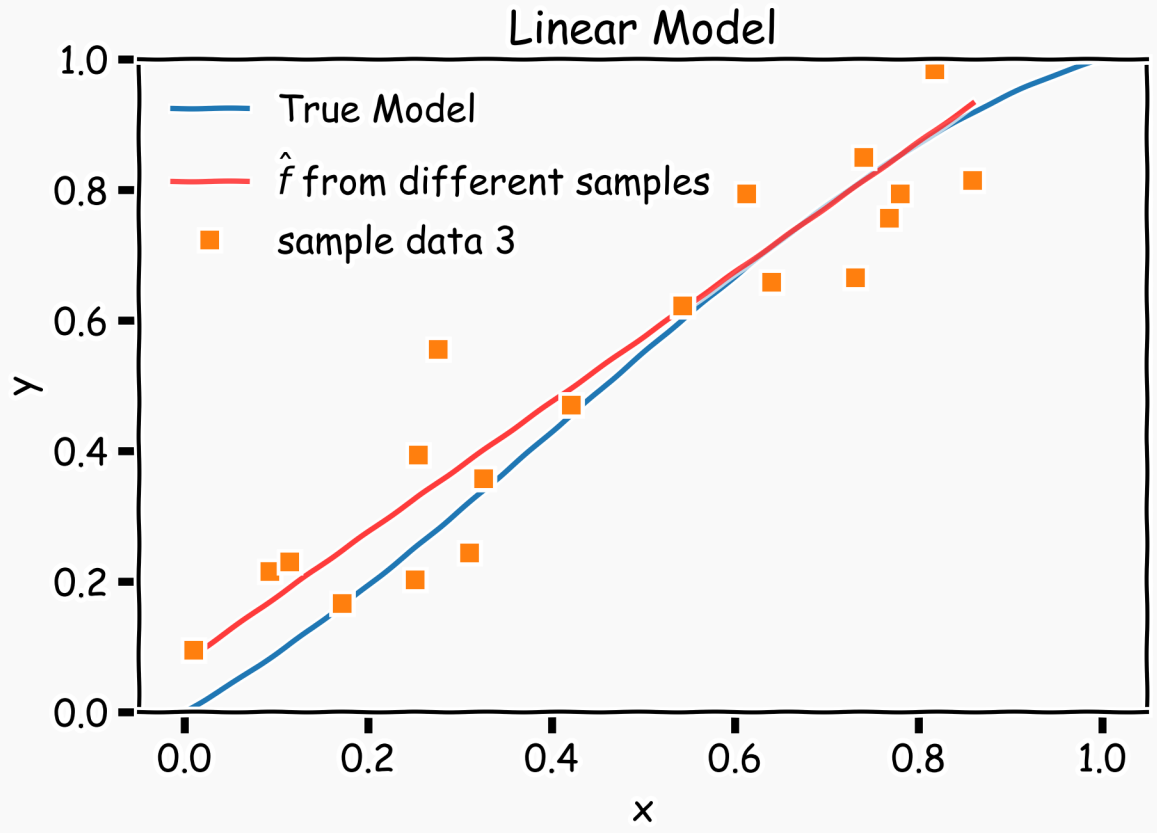
One more reason for overfitting:



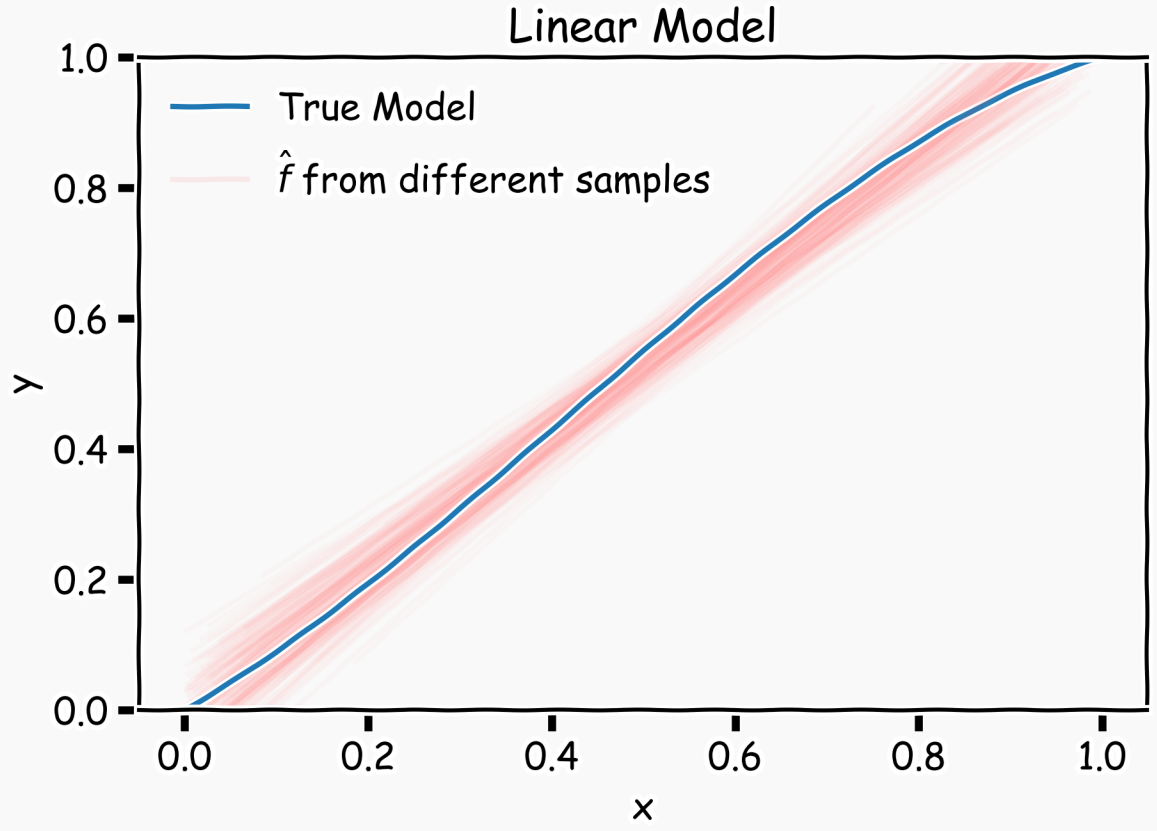
Bias vs Variance



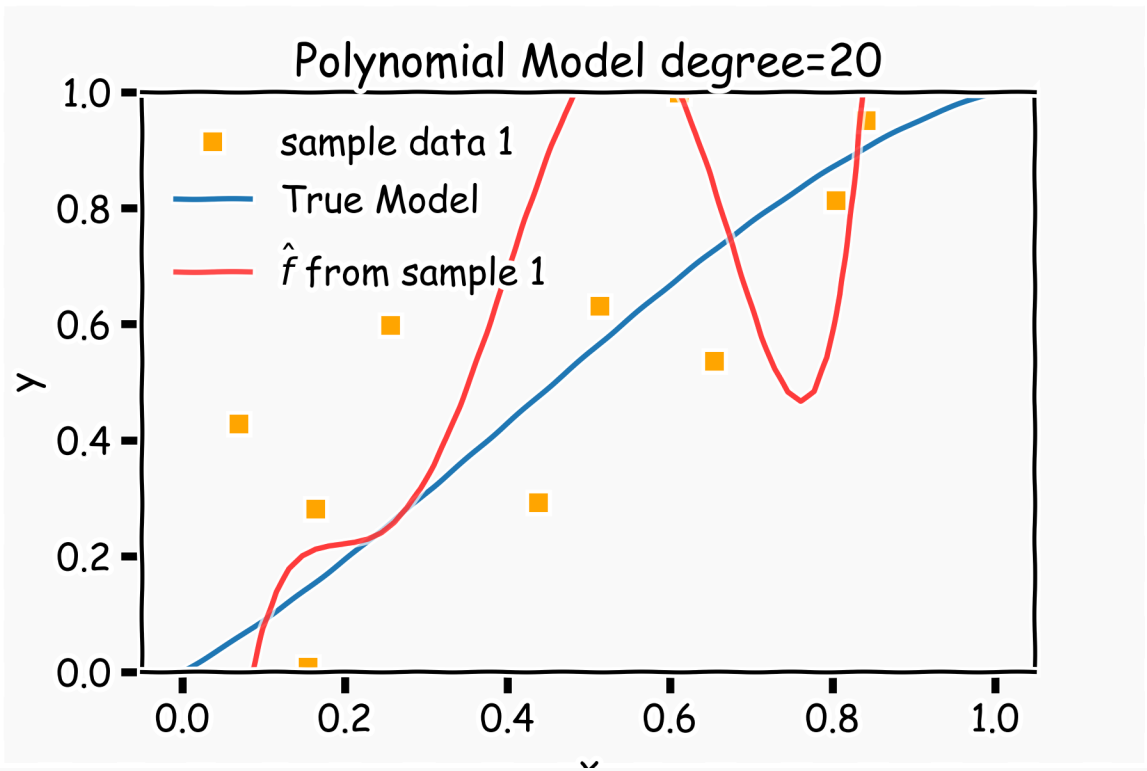
Bias vs Variance



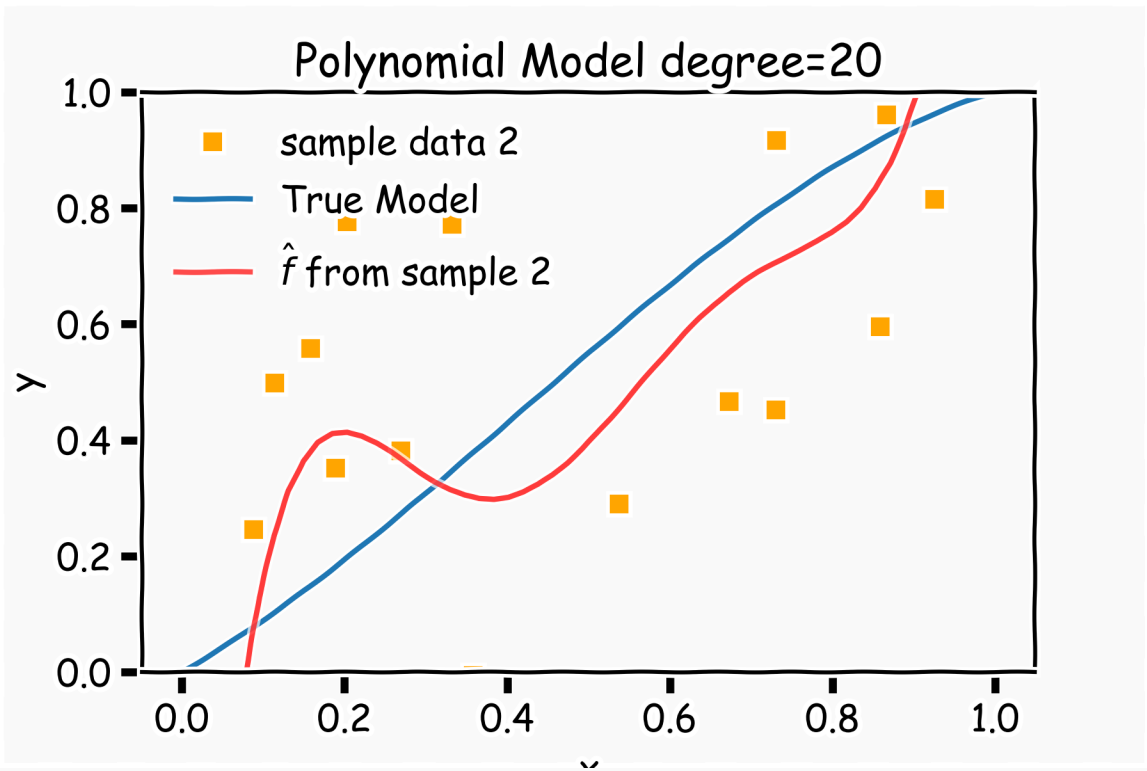
Linear models: 20 data points per line 2000 simulations



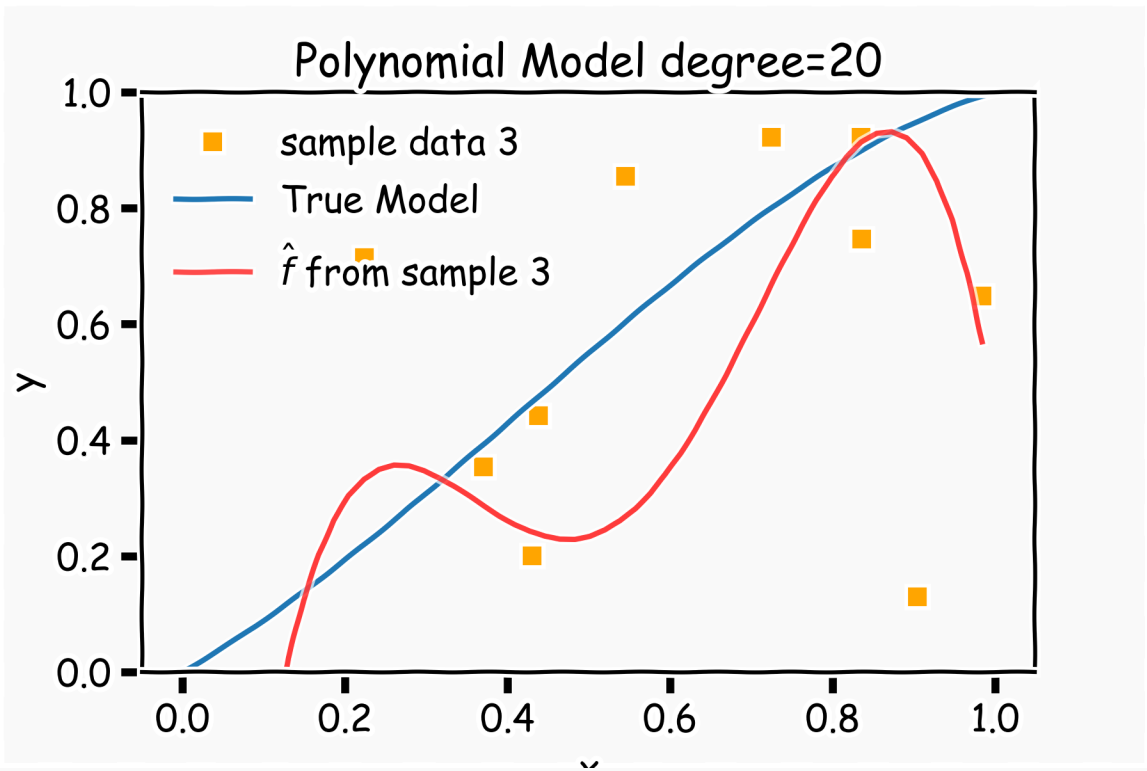
Bias vs Variance



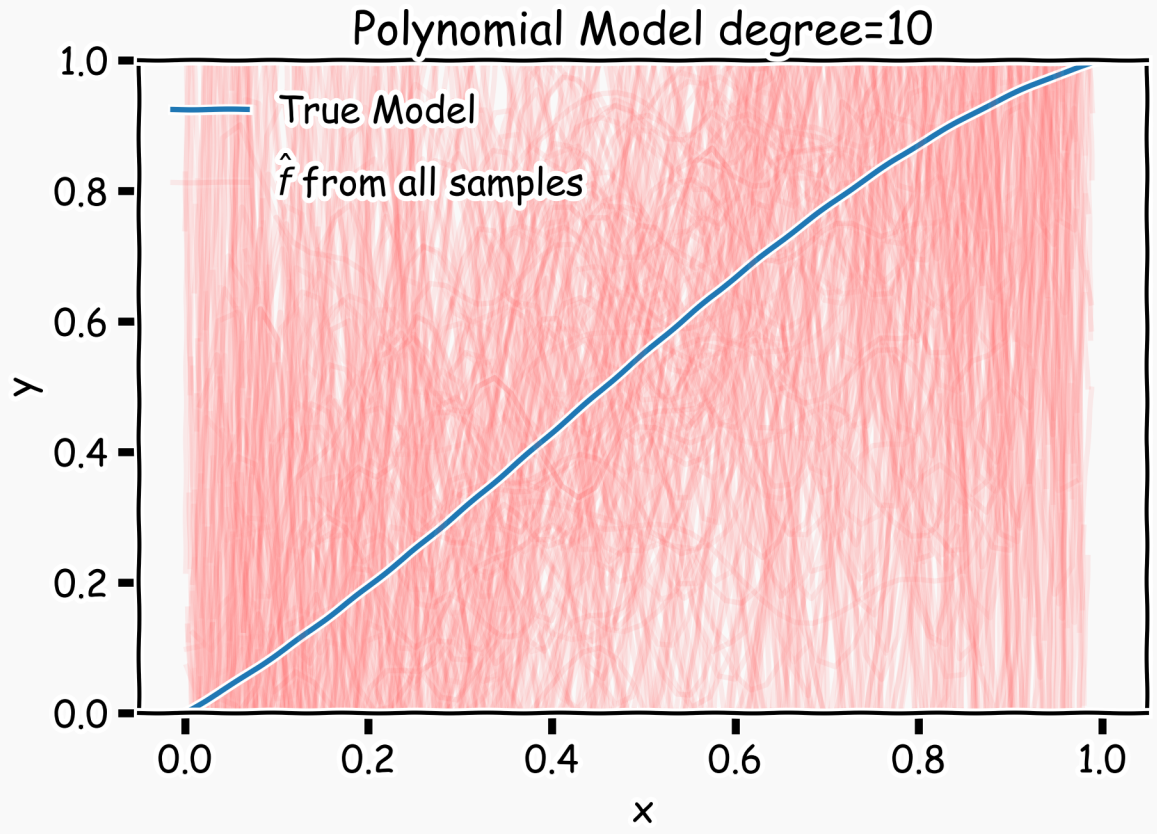
Bias vs Variance



Bias vs Variance



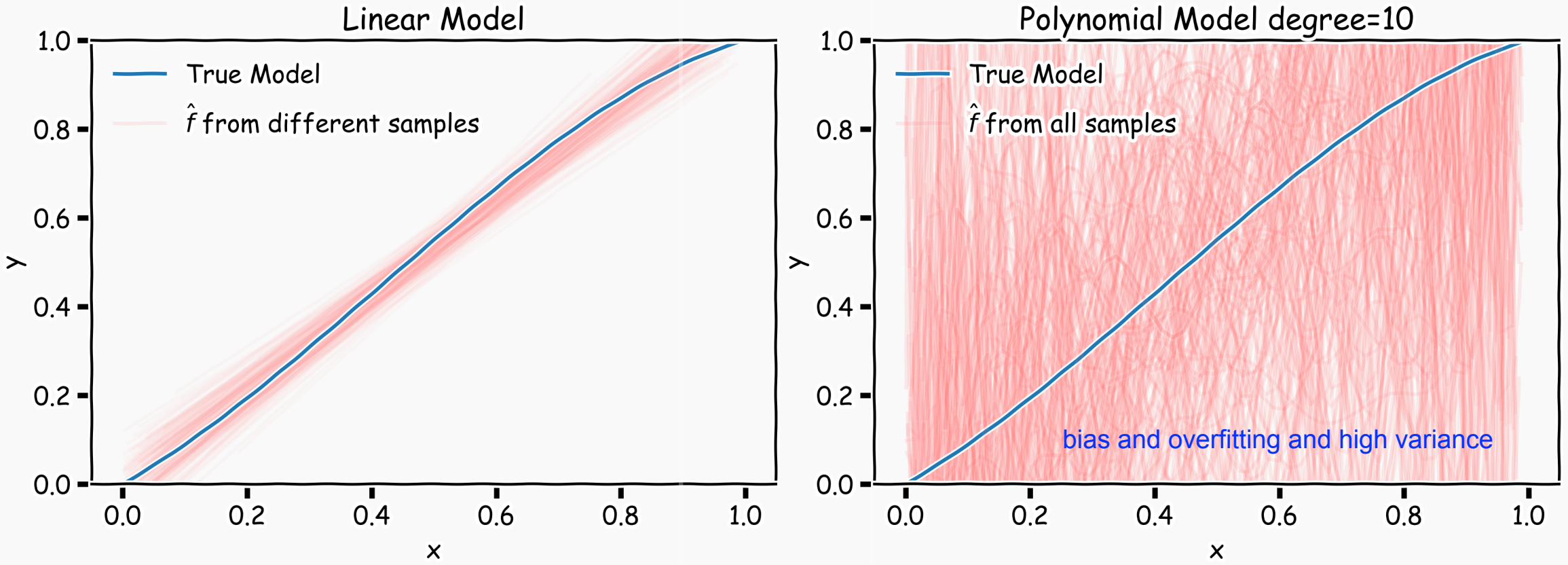
Poly 10 degree models : 20 data points per line 2000 simulations



Bias vs Variance

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

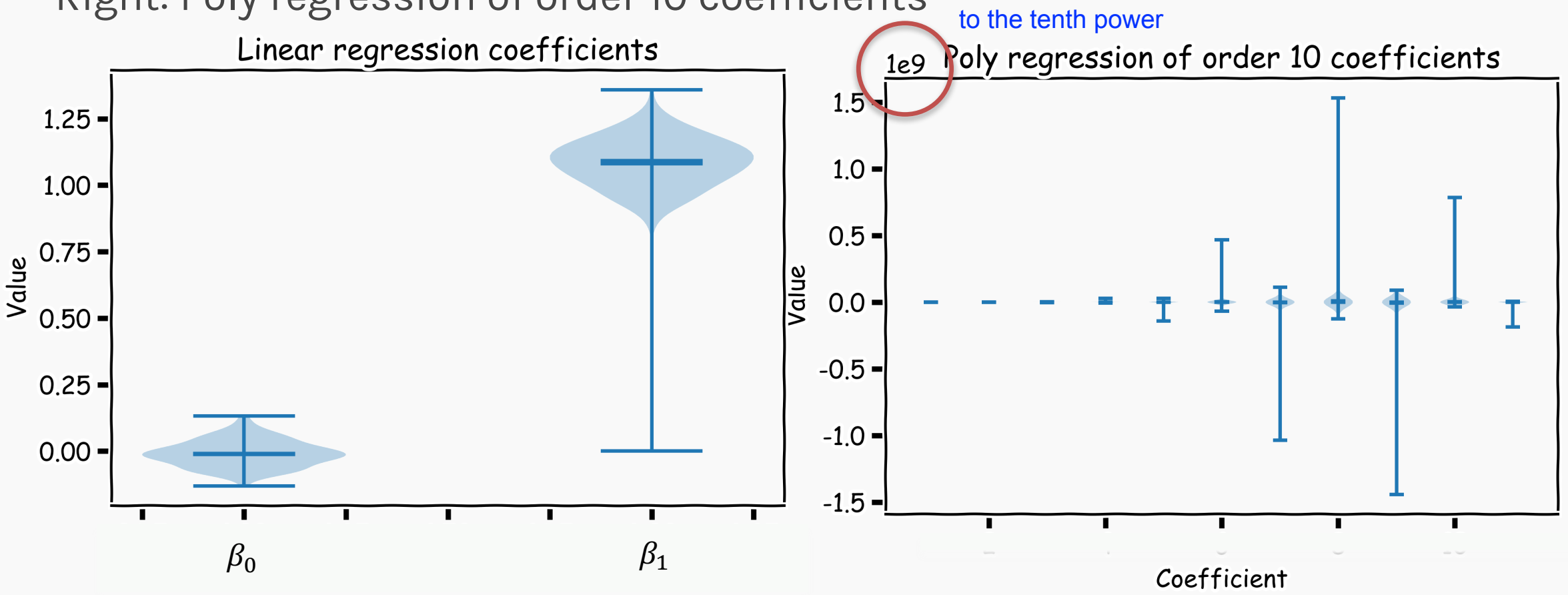
Right: Best-fit models using degree 10 polynomial



Bias vs Variance

Left: Linear regression coefficients

Right: Poly regression of order 10 coefficients



Lecture Outline

Overfitting

Model Selection

Cross Validation

Bias vs Variance

Regularization: LASSO and Ridge

Regularization Methods: A Comparison

Regularization: LASSO and Ridge

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:

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

MSE

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

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

LASSO Regression

Hence, we often say that L_{LASSO} is the loss function for l_1 regularization.

Finding the model parameters β_{LASSO} that minimize the l_1 regularized loss function is called ***LASSO regression***.

In [ ]:
```
from sklearn.linear_model import Lasso
```

In [22]:
```
lasso_regression = Lasso(alpha=1.0, fit_intercept=True)
lasso_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Lasso regression model:\n {} + {}^T . x'.format(lasso_regression.intercept_, lasso_regression.coe
```

```
Lasso regression model:
 10.424895873901445 + [ 0.24482603  3.48164594  1.84836859 -0.06864603 -0.         -0.
 -0.02249766 -0.          0.          0.          0.          ]^T . x
```

In [23]:
```
print('Train R^2: {}, test R^2: {}'.format(lasso_regression.score(np.vstack((X_train, X_val)),
                                                                  np.hstack((y_train, y_val))),
                                           lasso_regression.score(X_test, y_test)))
```

```
Train R^2: 0.48154992527975765, test R^2: 0.6846451270316087
```

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 $\boldsymbol{\beta}$

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

Ridge Regression

Hence, we often say that L_{ridge} is the loss function for l_2 regularization.

Finding the model parameters β_{ridge} that minimize the l_2 regularized loss function is called ***ridge regression.***

In [ ]:
```
from sklearn.linear_model import Ridge
```

In [20]:
```
X_train = train[all_predictors].values
X_val = validation[all_predictors].values
X_test = test[all_predictors].values

ridge_regression = Ridge(alpha=1.0, fit_intercept=True)
ridge_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Ridge regression model:\n {} + {}^T . x'.format(ridge_regression.intercept_, ridge_regression.coe
```

```
Ridge regression model:
 -525.7662550875951 + [ 0.24007312  8.42566029  2.04098593 -0.04449172 -0.01227935  0.41902475
 -0.50397312 -4.47065168  4.99834262  0.          0.          0.29892679]^T . x
```

In [21]:
```
print('Train R^2: {}, test R^2: {}'.format(ridge_regression.score(np.vstack((X_train, X_val)),
                                                                  np.hstack((y_train, y_val))),
                                           ridge_regression.score(X_test, y_test)))
```

```
Train R^2: 0.5319764744847737, test R^2: 0.7881798111697319
```

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 $\boldsymbol{\beta}$,

- 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 $\boldsymbol{\beta}_{\text{ridge}}$ and $\boldsymbol{\beta}_{\text{LASSO}}$ to be close to zero.

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

Ridge - Computational complexity

Solution to ridge regression:

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

The solution of the Ridge/Lasso regression involves three steps

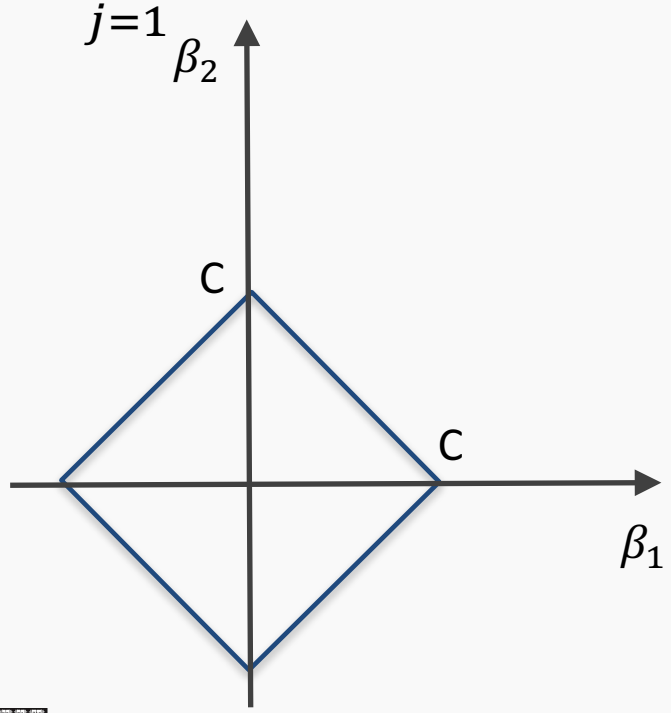
- Select λ
- Find the minimum of the ridge/Lasso regression cost function (using linear algebra) as with the multiple regression and record the R^2 **on the test set**.
- Find the λ that gives the largest R^2

The Geometry of Regularization (LASSO)

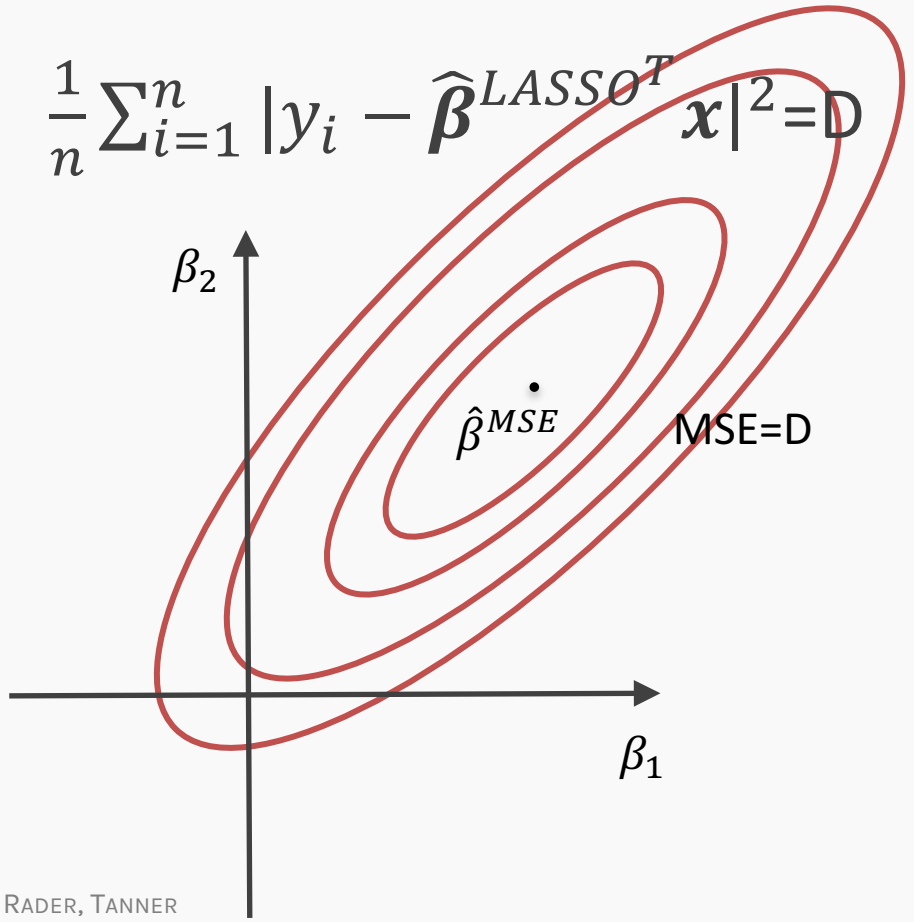
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

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

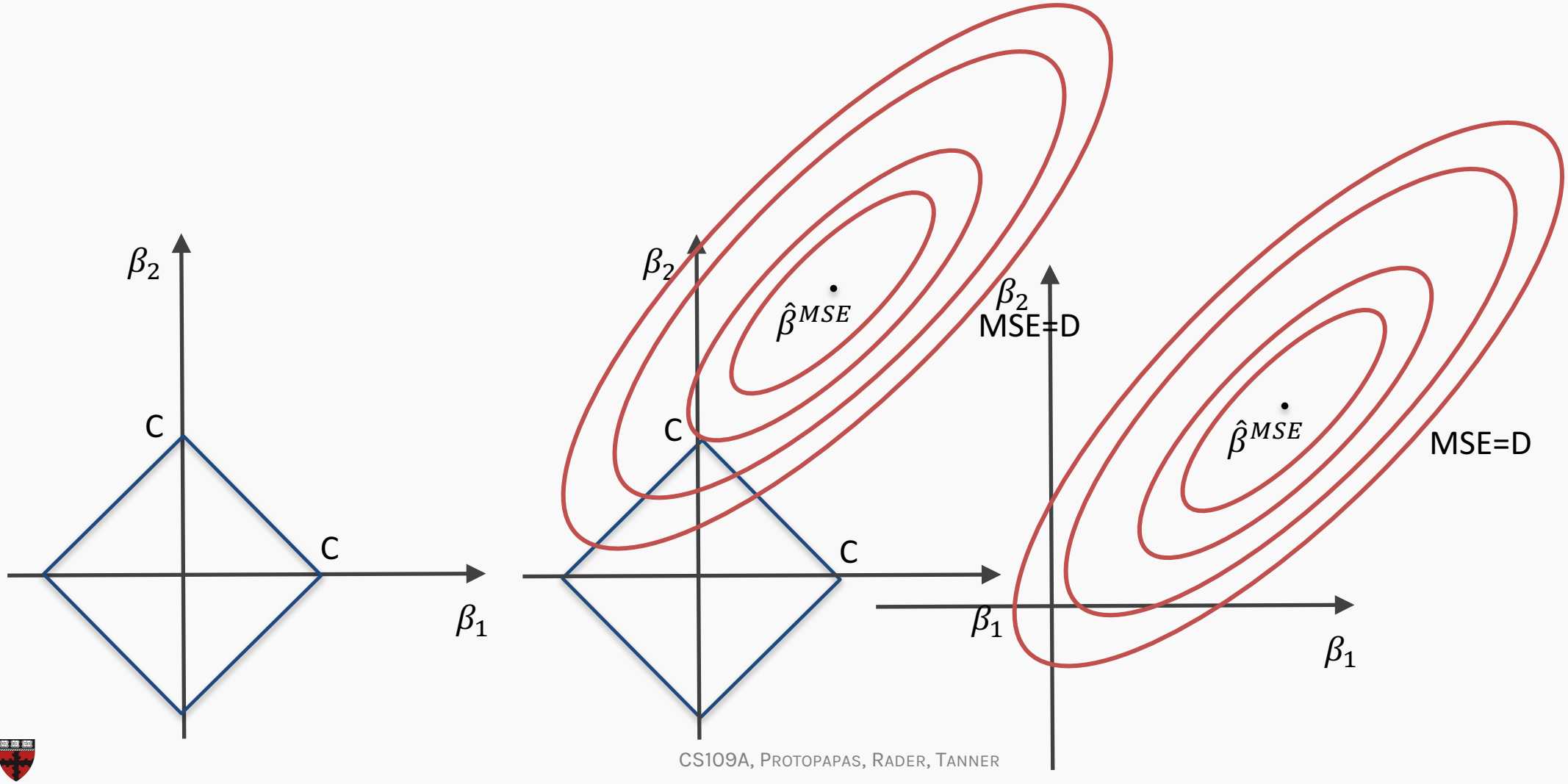
$$\lambda \sum_{j=1}^{J} |\hat{\beta}_j^{LASSO}| = C$$



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



The Geometry of Regularization (LASSO)

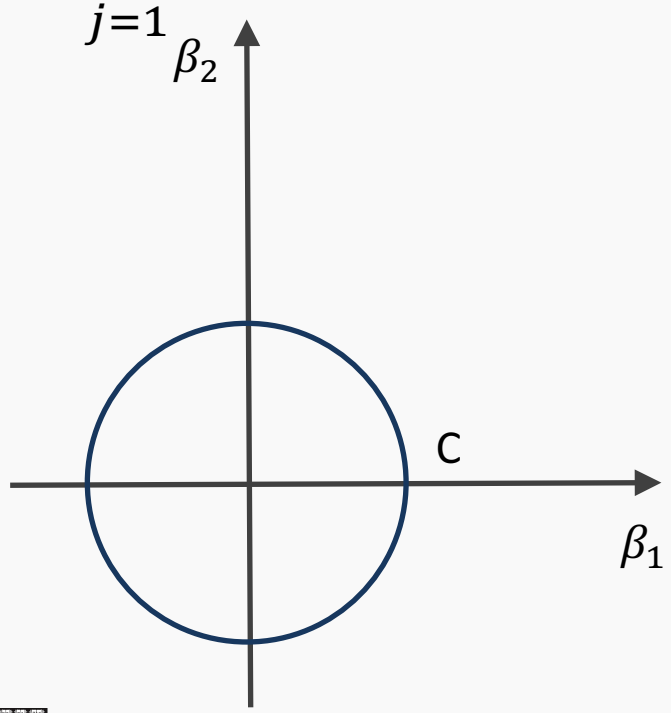


The Geometry of Regularization (Ridge)

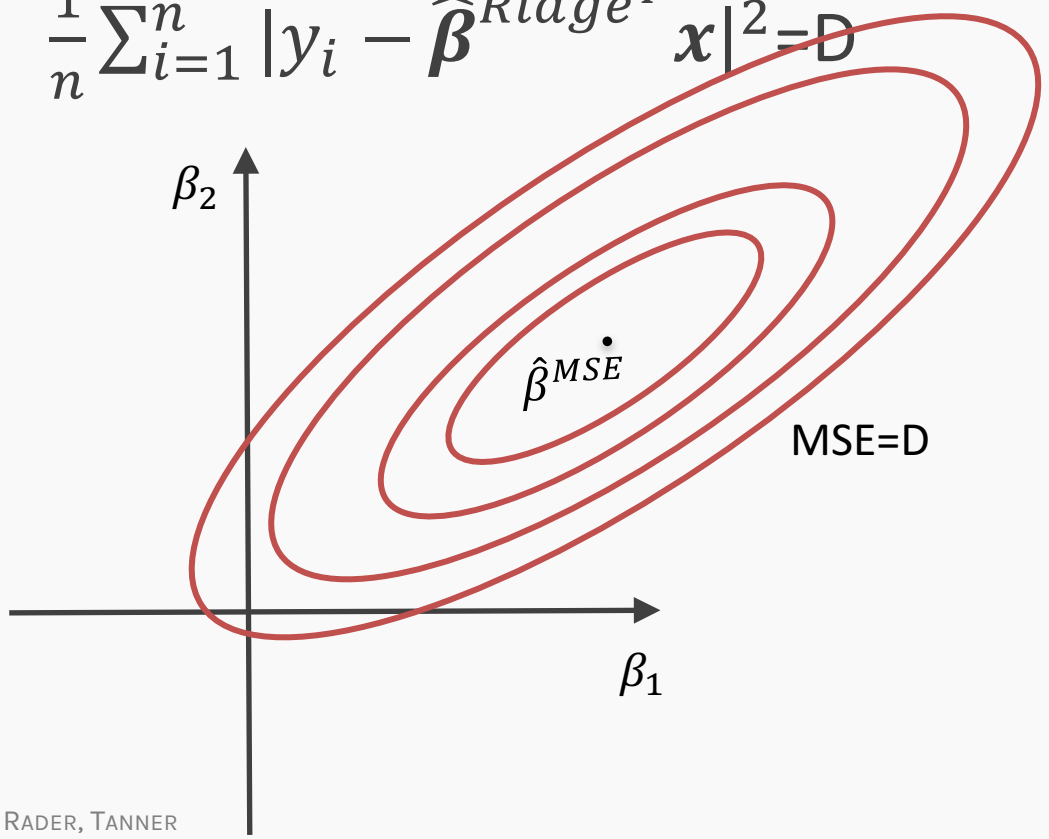
$$L_{Ridge}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \boldsymbol{x}\Big|^2 + \lambda \sum_{j=1}^{J} (\beta_j)^2$$

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

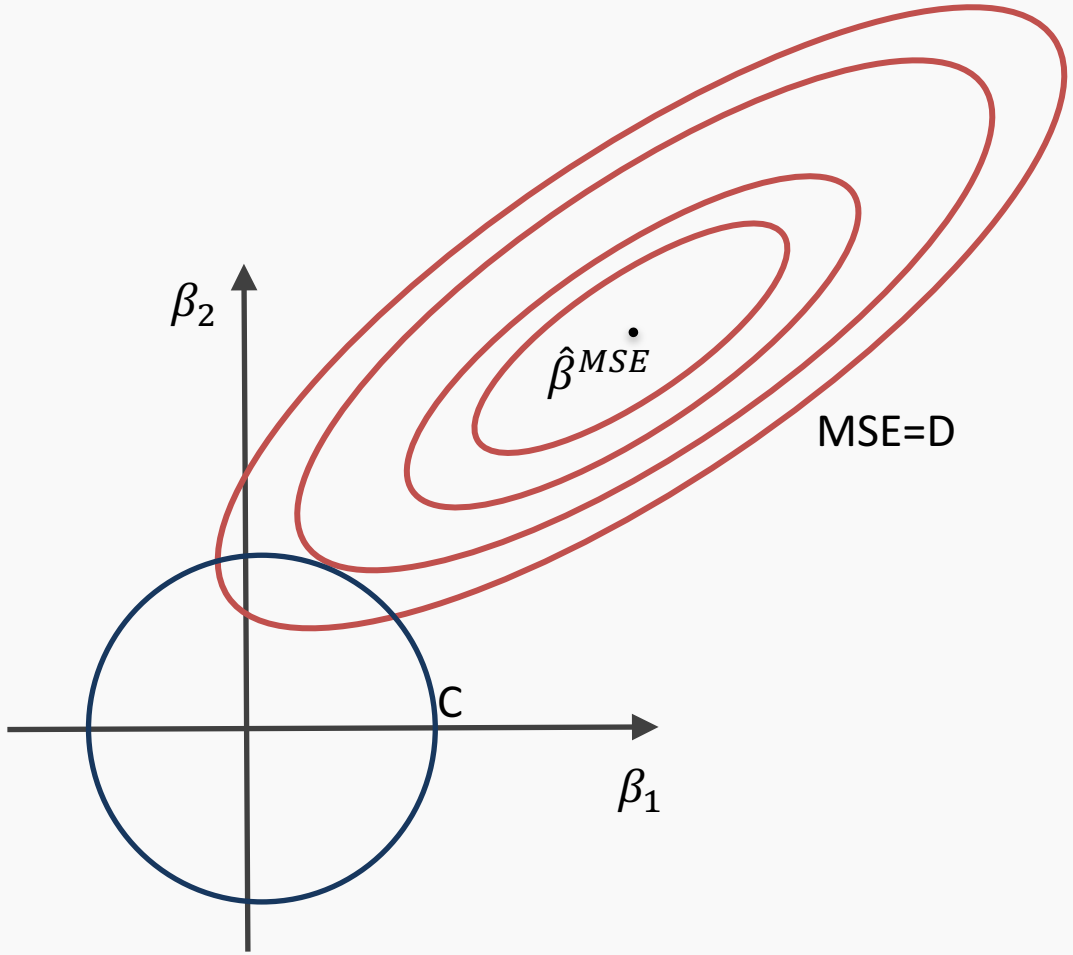
$$\lambda \sum_{j=1}^{J} \Big|\hat{\beta}_j^{Ridge}\Big|^2 = C$$



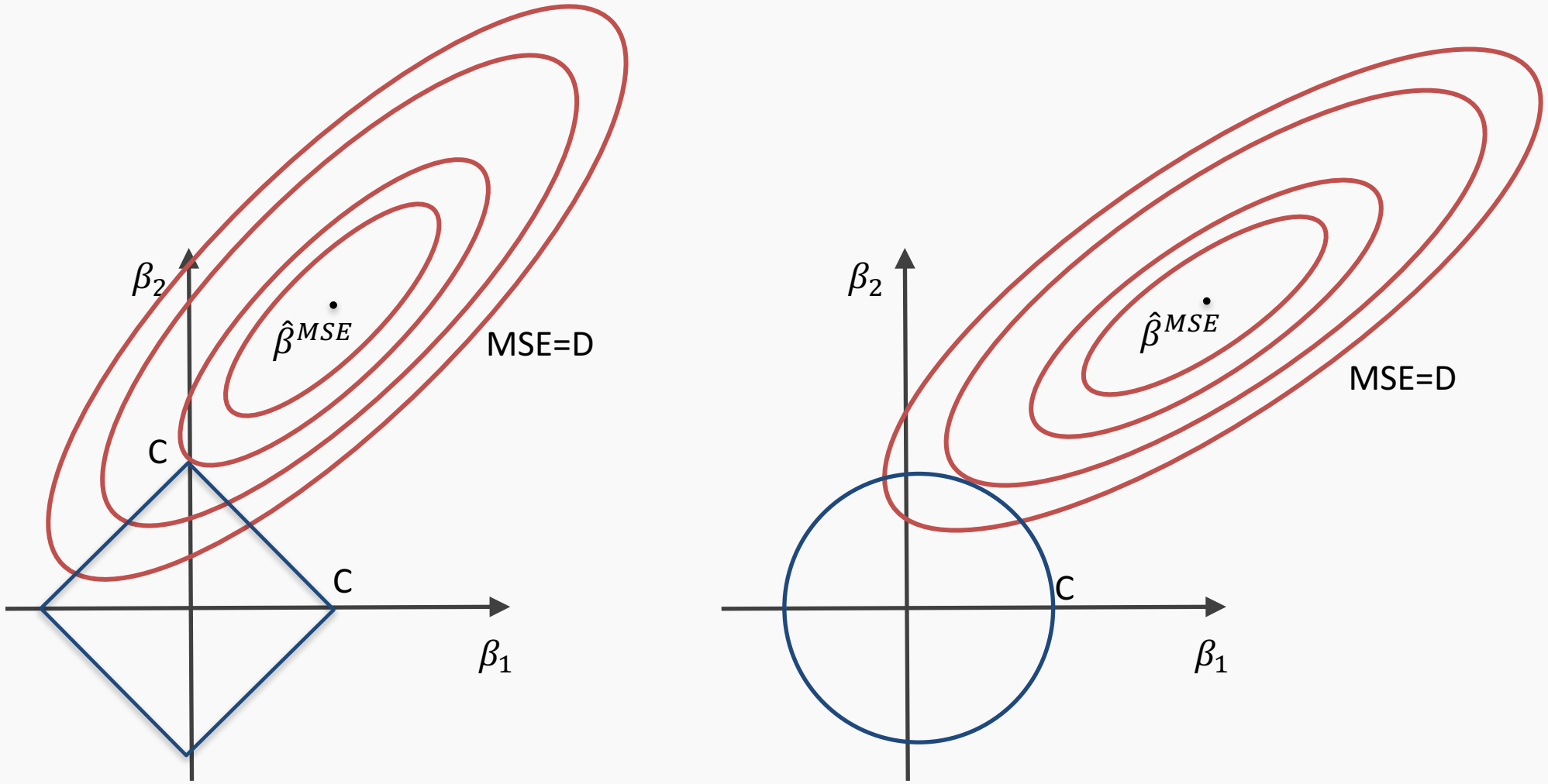
$$\frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{\boldsymbol{\beta}}^{Ridge^T} \boldsymbol{x}|^2 = D$$



The Geometry of Regularization (Ridge)



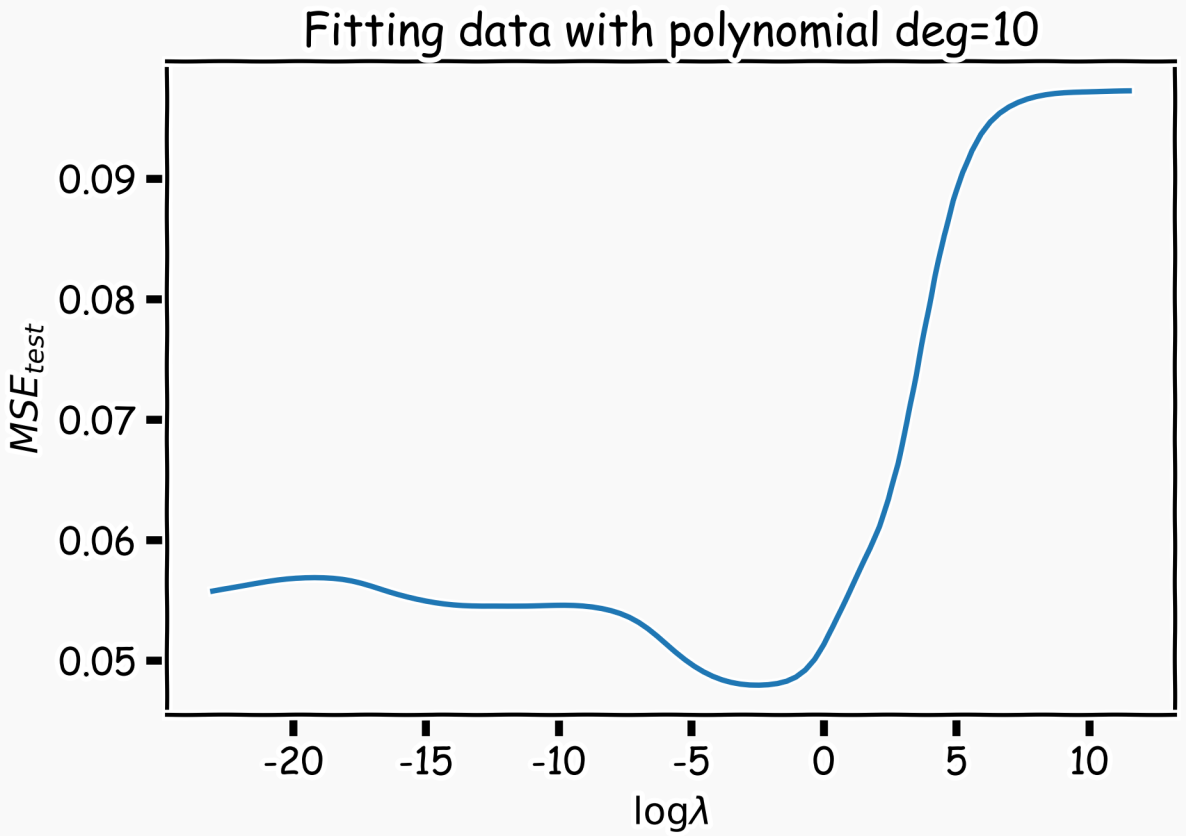
The Geometry of Regularization



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

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \ldots \lambda_{max}\}$:
    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

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 $\hat{\beta}_{lasso}(\lambda_{lasso})$

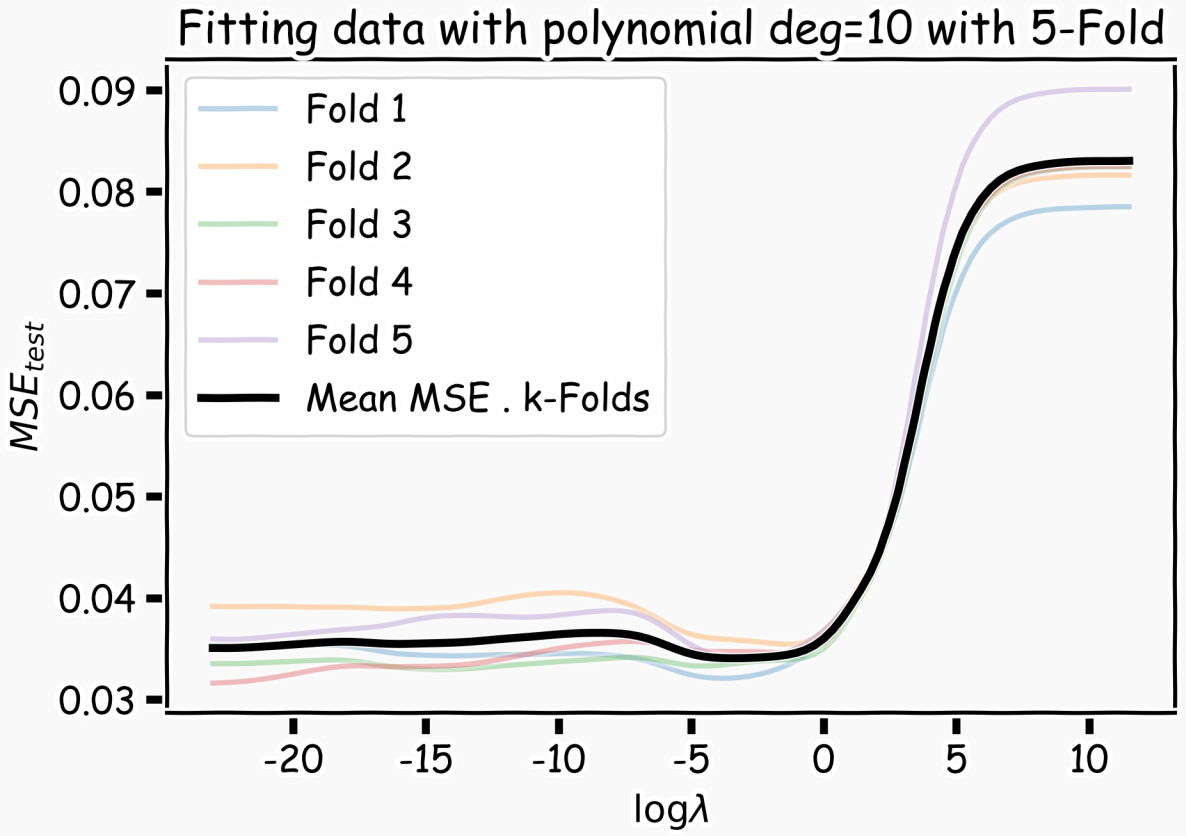
Ridge regularization with **CV**: step by step

	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 $\{1, \ldots, K\}$
    1. for λ in $\{\lambda_0, \ldots, \lambda_n\}$:
        A. determine the β that minimizes the L_{ridge}, $\hat{\beta}_{ridge}(\lambda, k) = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda I\right)^{-1} X^T Y$, 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



Variable Selection as Regularization

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

Many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) rather than stepwise selection, as LASSO avoids the statistic problems that arises in stepwise selection.

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

Afternoon Exercises

Quiz - to be completed in the next 10 min:

Sway: Lecture 7

Programmatic – to be completed by Lab tomorrow:

Lessons: Lecture 7:


Tableau
66.6K views
0:01 / 1:29