Lecture 7: Regularization

CS109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader

Announcements

Sections: Due to low attendance, Monday evening section is canceled. Instead we add one extra OH on Wednesday 5:30 – 7 (high demand).

Office Hours for 209: You can use Pavlos and Kevin's OH for 209 material.

Homeworks:

- Minor correction in the language for HW3.
- HW4 will be released on Tuesday 11:59pm.
    - Two weeks
    - individual
    - Piazza only private messages.

Projects:

- Milestone 1 due on Oct 3

Lecture Outline

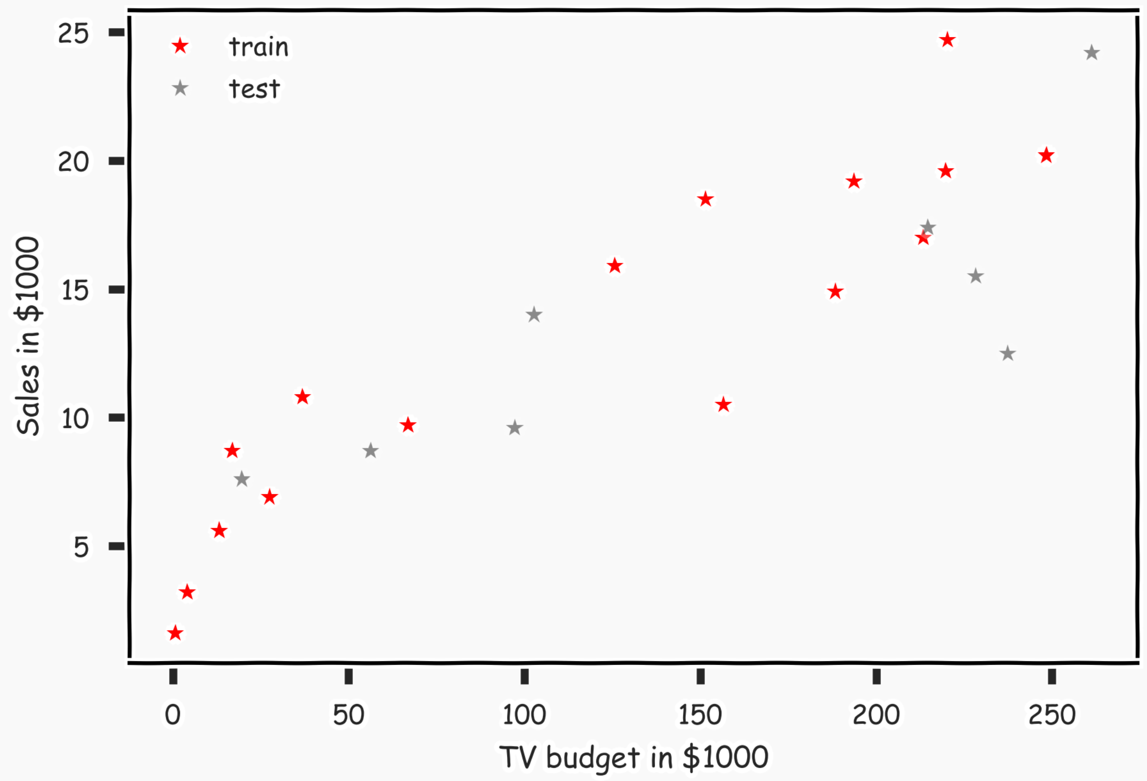
Overfitting

Bias vs Variance

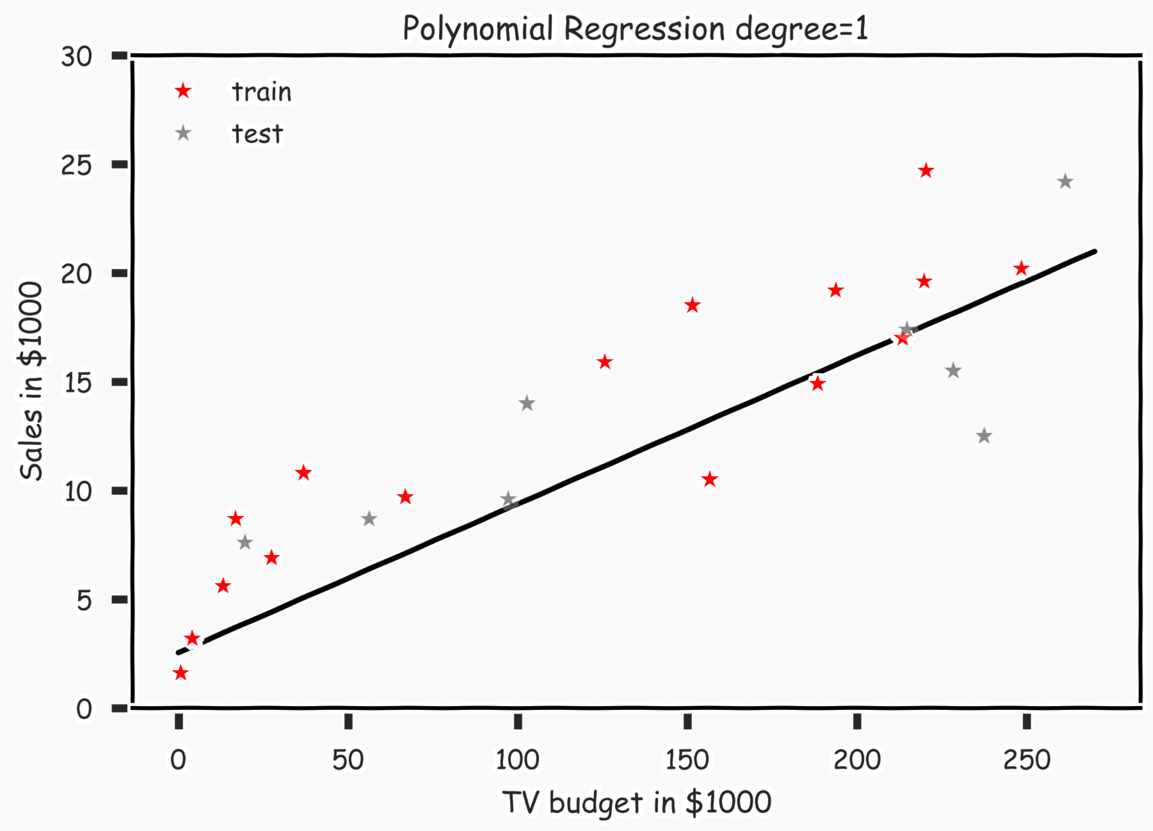
Regularization: LASSO and Ridge

Regularization Methods: A Comparison

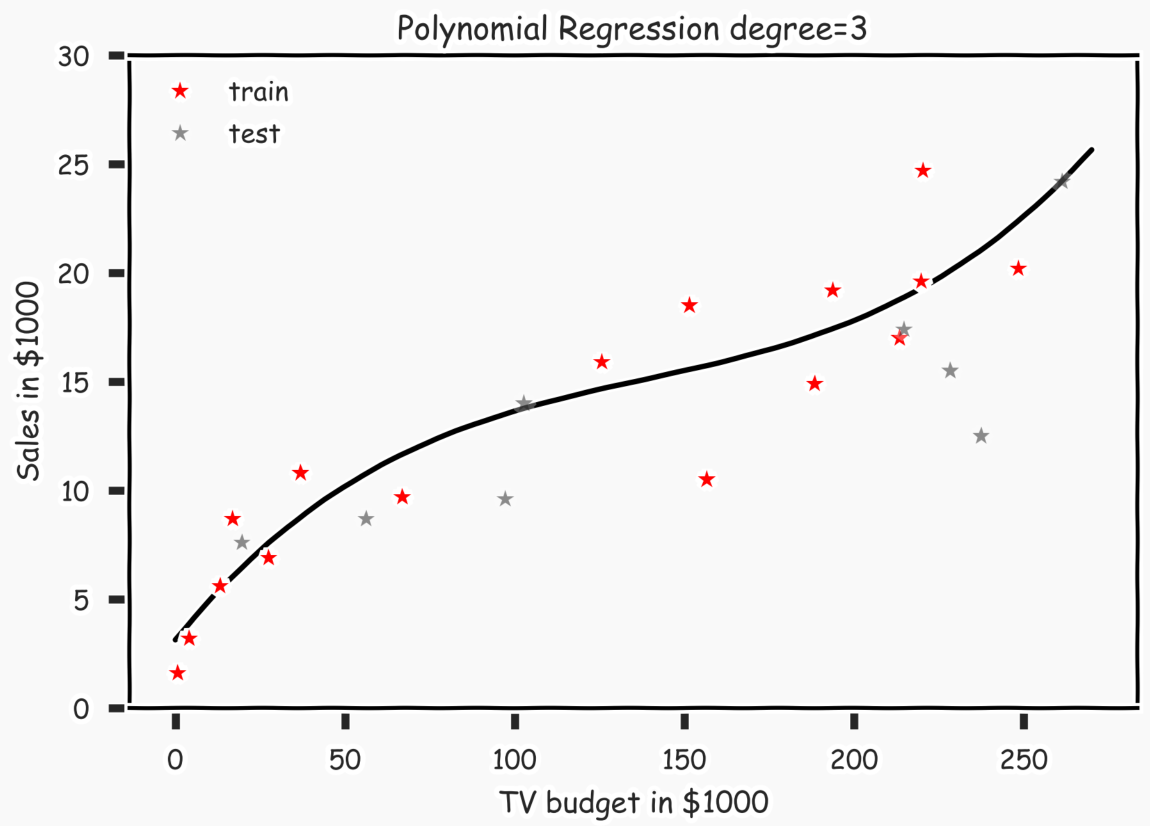
Overfitting



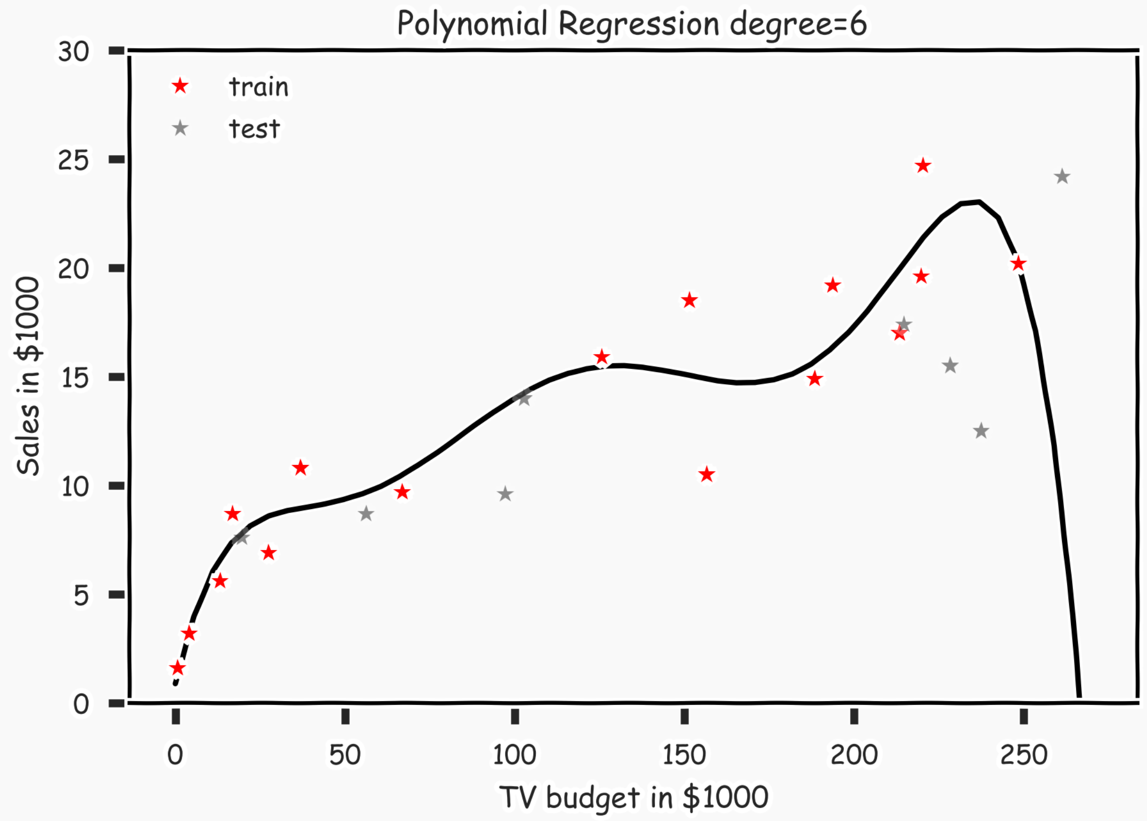
Overfitting



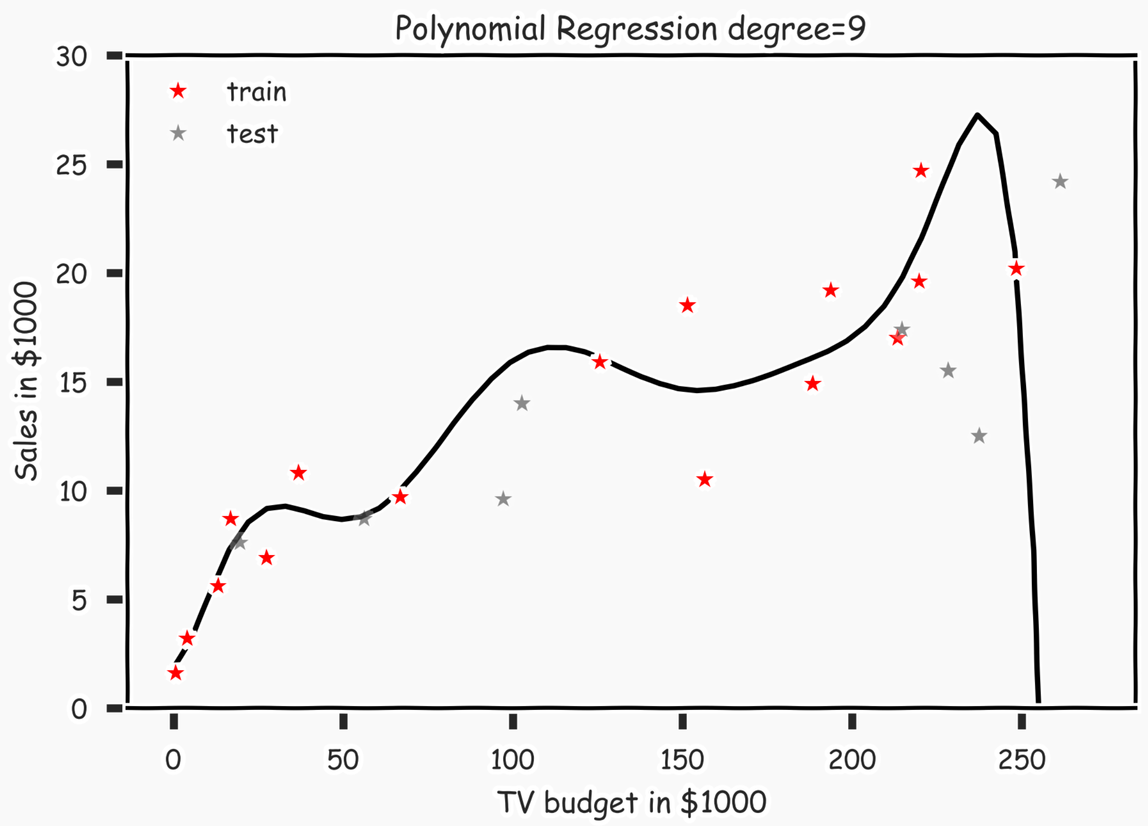
Overfitting



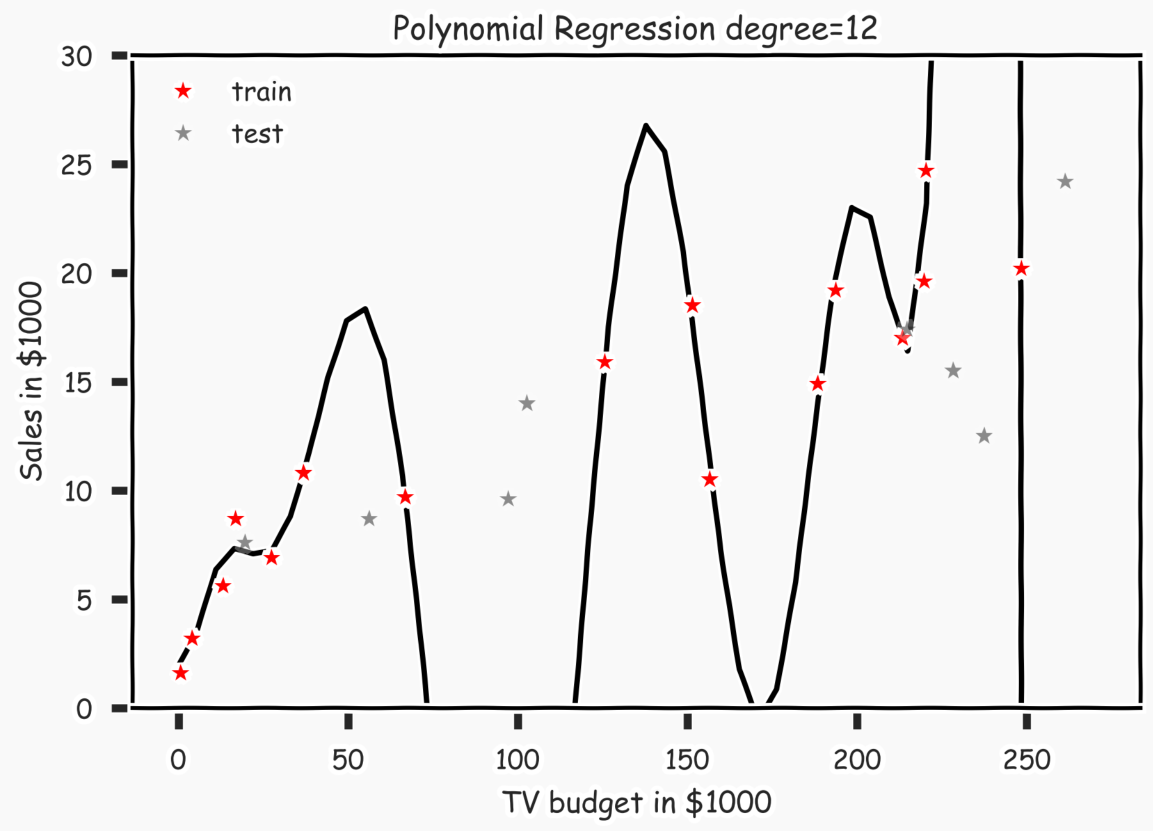
Overfitting



Overfitting



Overfitting



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**

Stepwise Variable Selection and Cross Validation

Last time, we addressed the issue of 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,

Today we will see selecting an optimal model by cross validation - evaluating each model on multiple validation sets.

And also today, we will also address the issue of discouraging extreme values in model parameters

Stepwise Variable Selection Computational Complexity

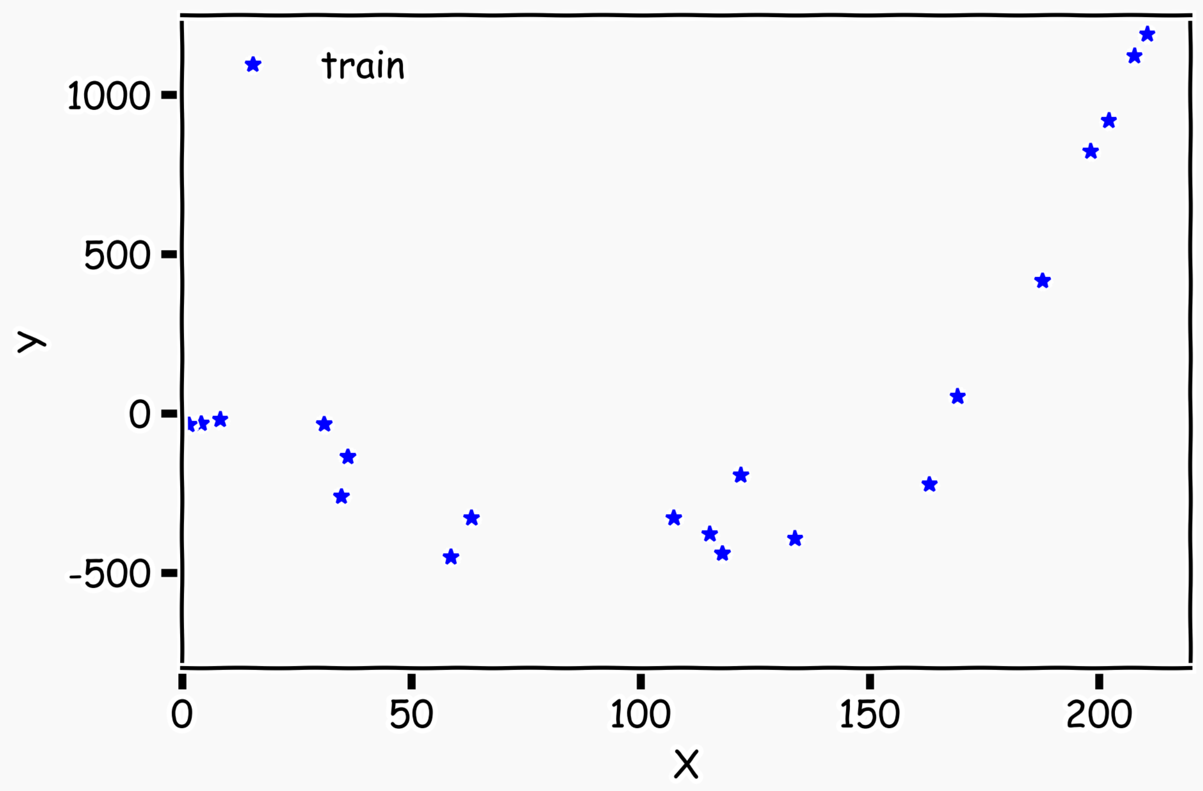
How many models did we evaluate?

- 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)
- …

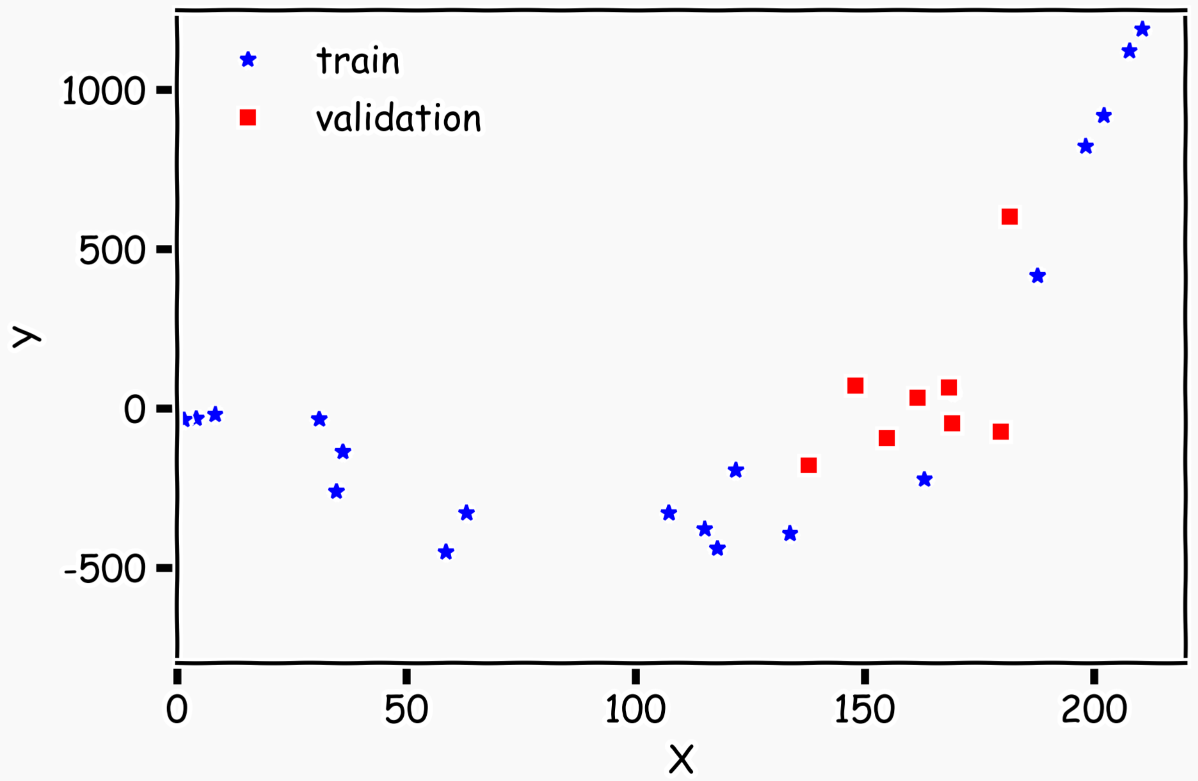
$$O(J^2) \ll 2^J \text{ for large } J$$

Cross Validation

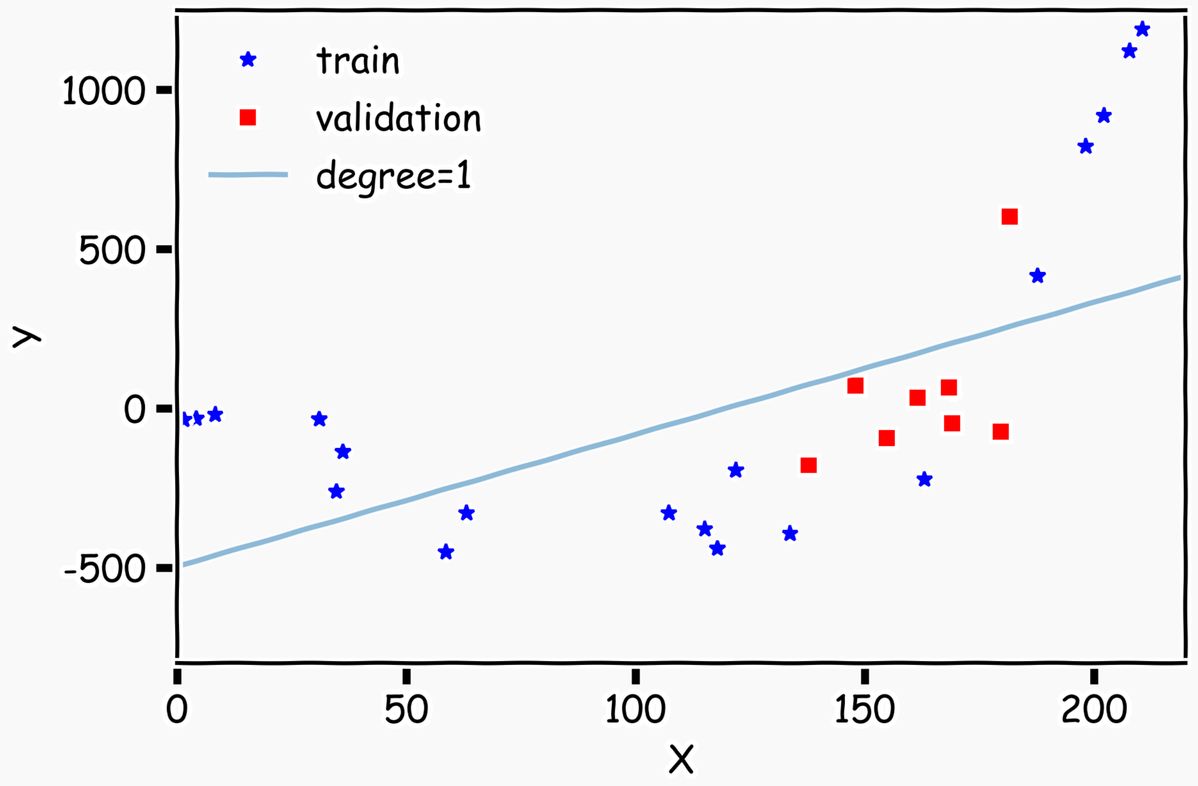
Cross Validation



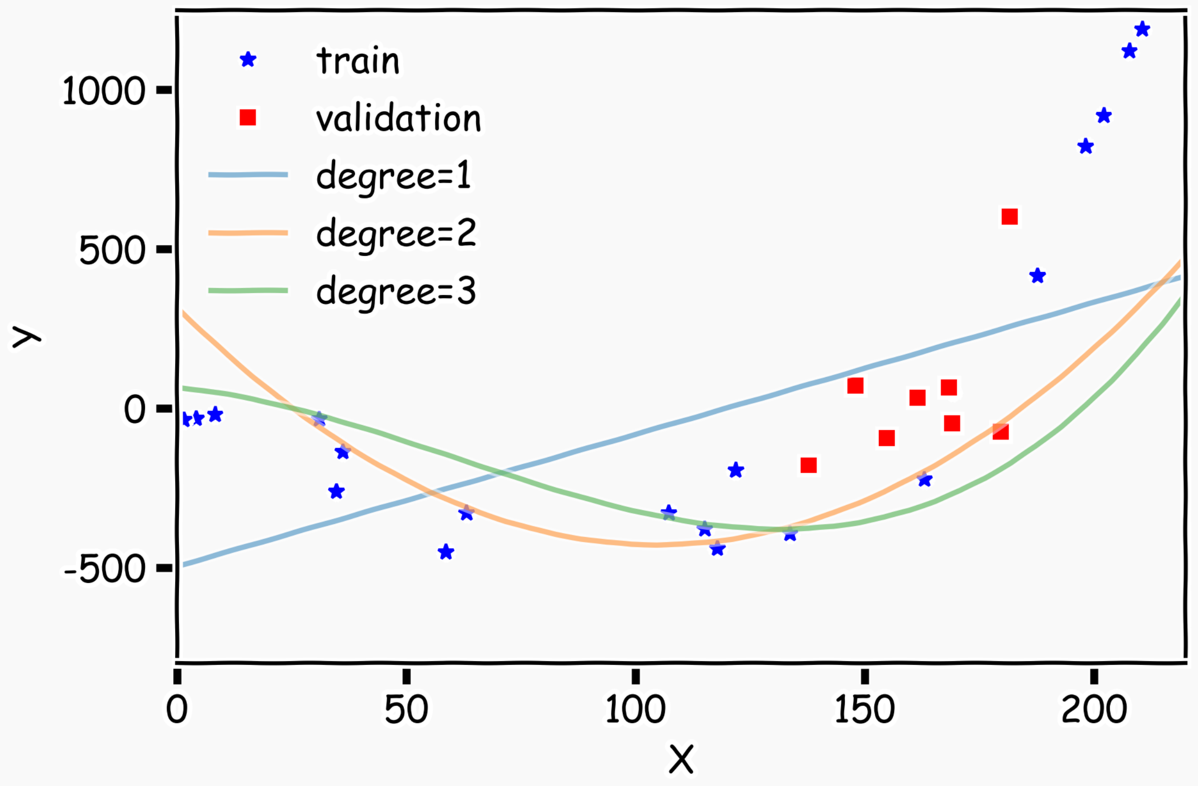
Cross Validation



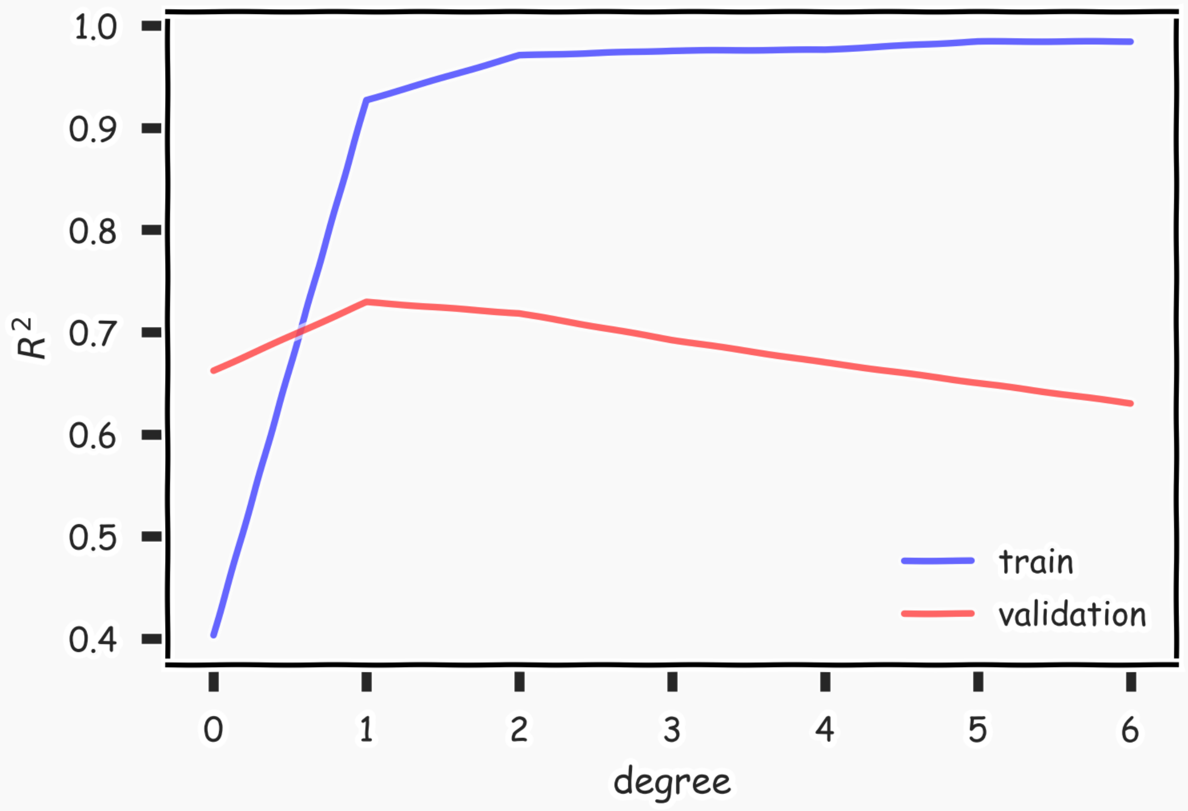
Cross Validation



Cross Validation



Validation



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.

Leave-One-Out

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 create training/validation splits 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.

K-Fold Cross Validation

Rather than creating *n* number of training/validation splits, each time leaving one data point for the validation set, we can include more data in the validation set using **K-fold validation**:

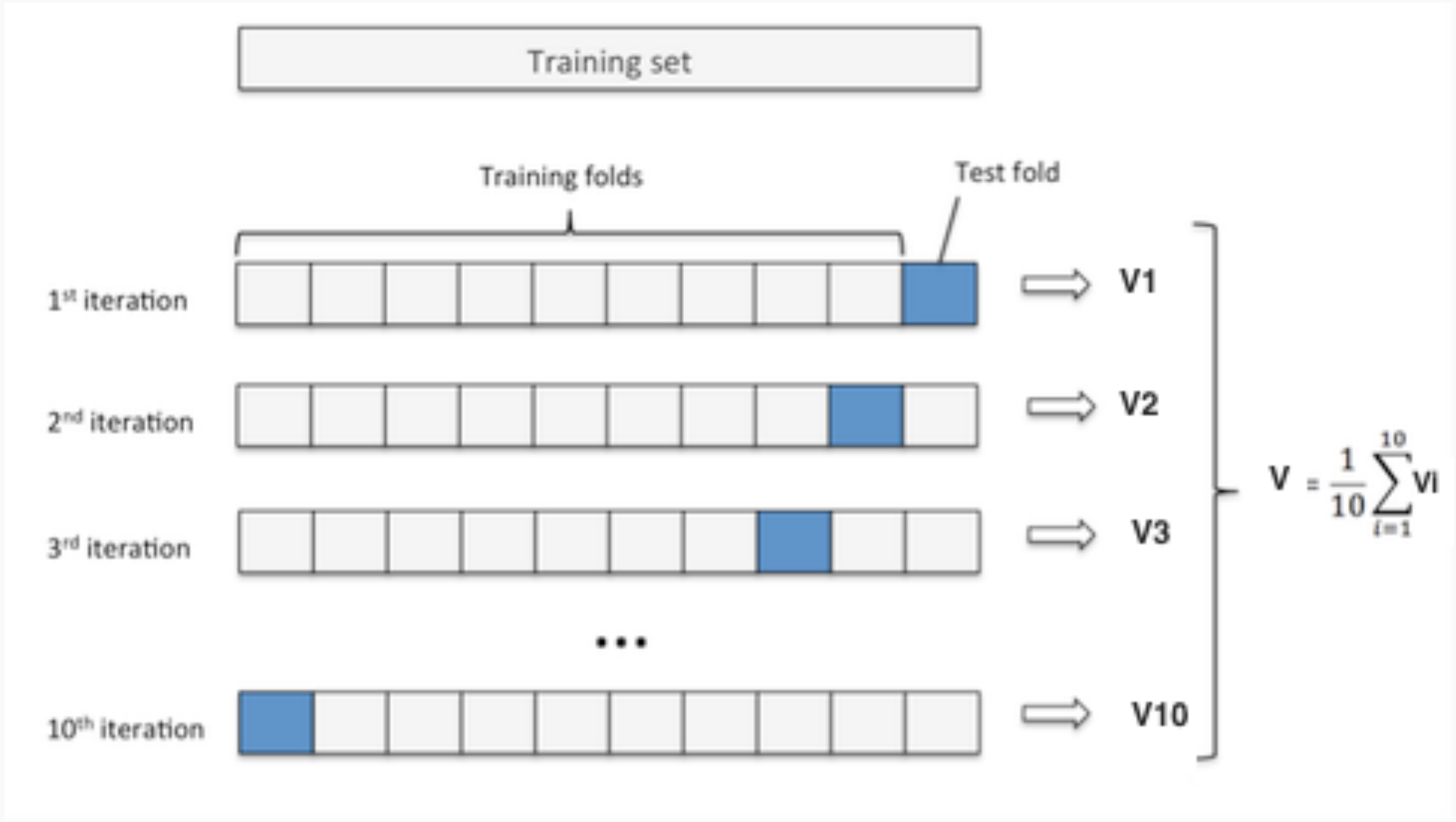
- split the data into *K* uniformly sized chunks, $\{C_1, \ldots, C_K\}$
- we create *K* number of training/validation splits, using one of the *K* chunks for validation and the rest for training.

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.

Cross Validation



Predictor Selection: Cross Validation

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

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

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

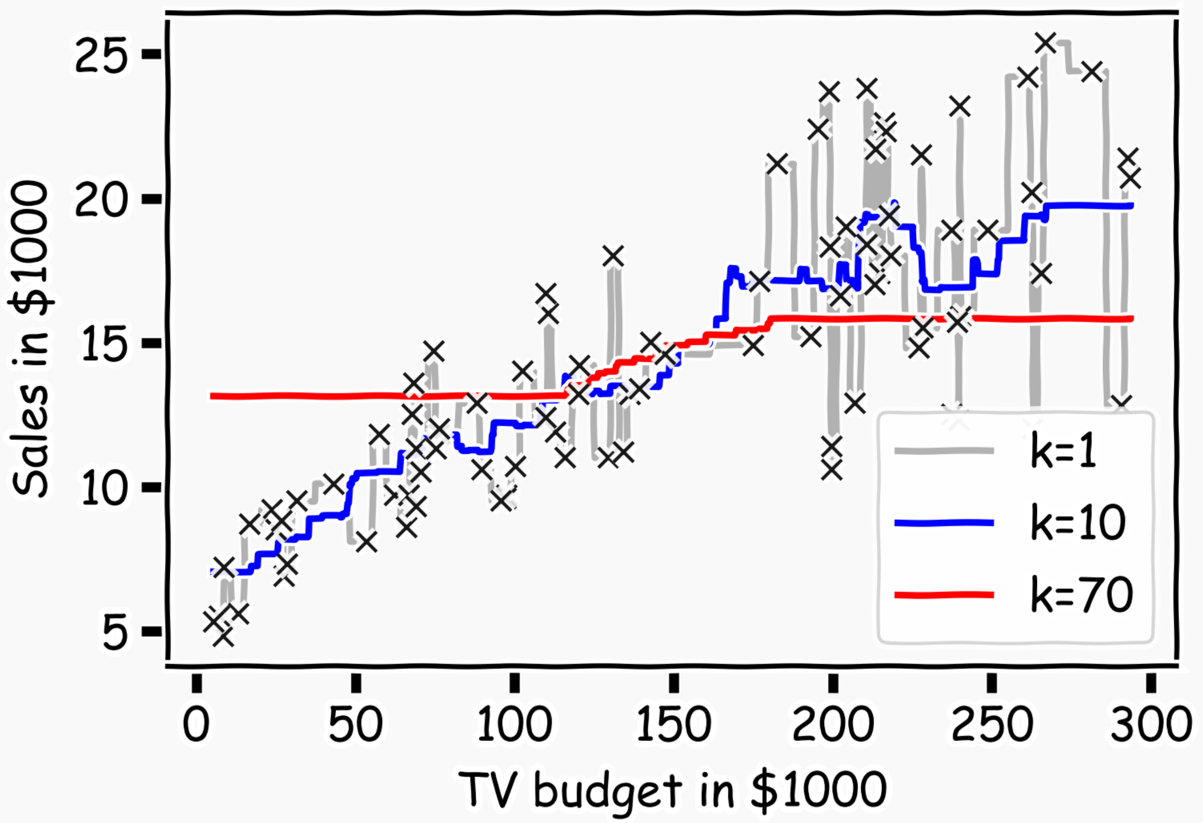
which of the predictors $\{x, x^2, \dots, x^m\}$, should we select for modeling?

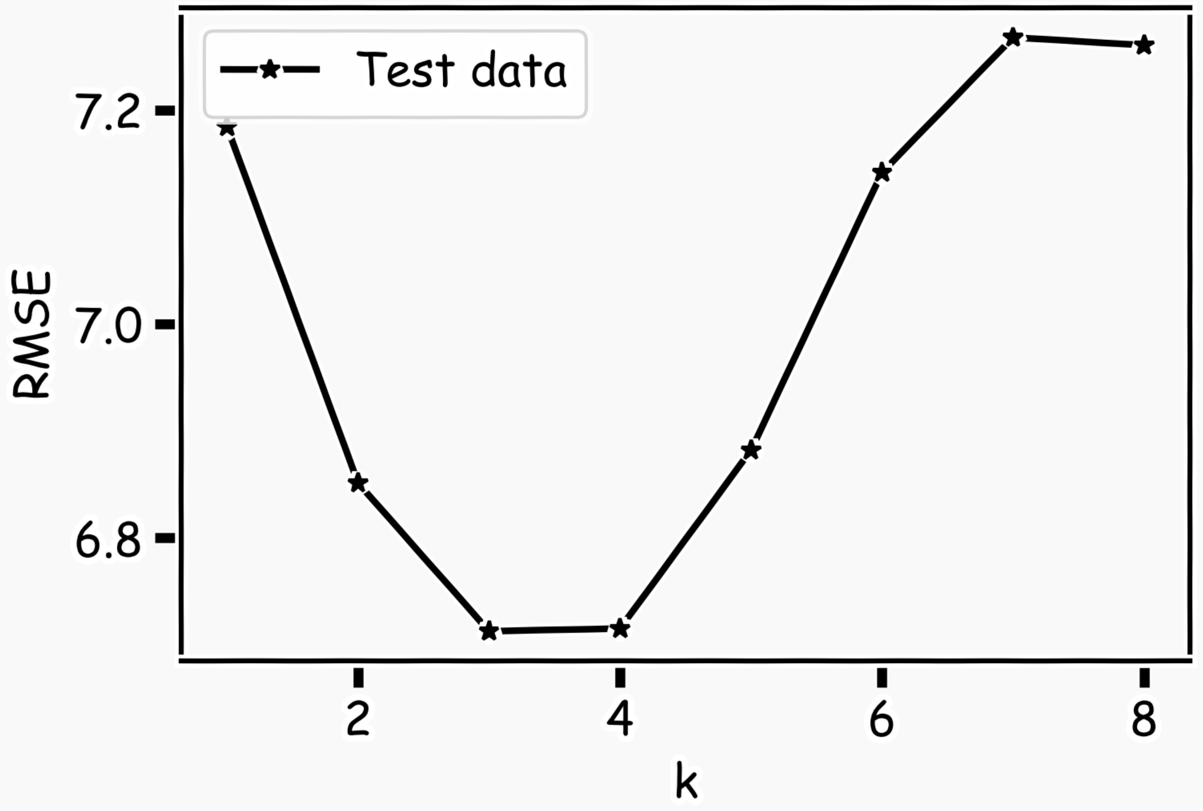
kNN Revisited

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.

kNN Revisited





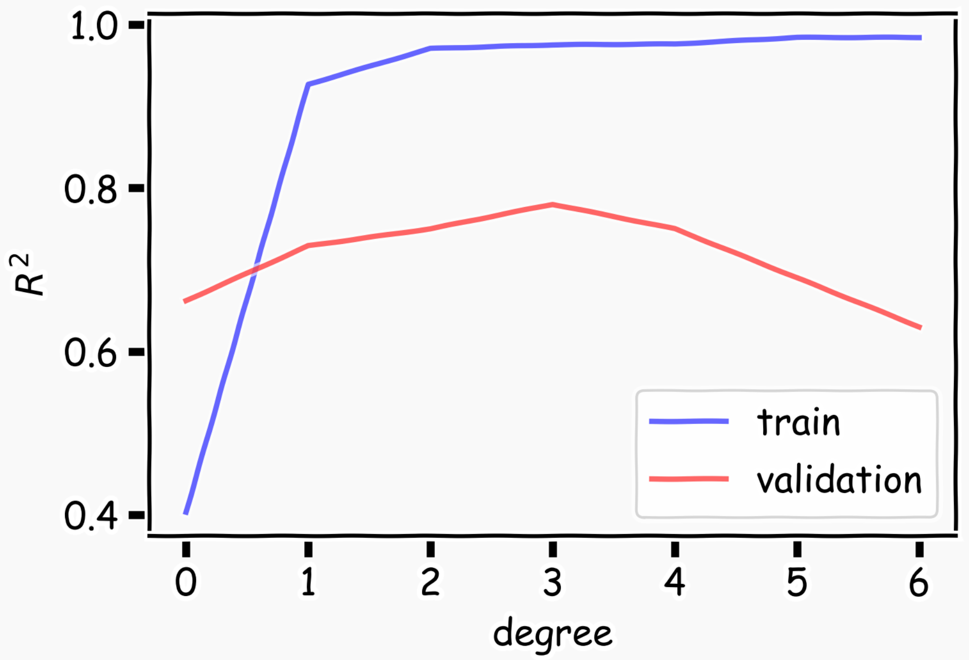
kNN Revisited

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.

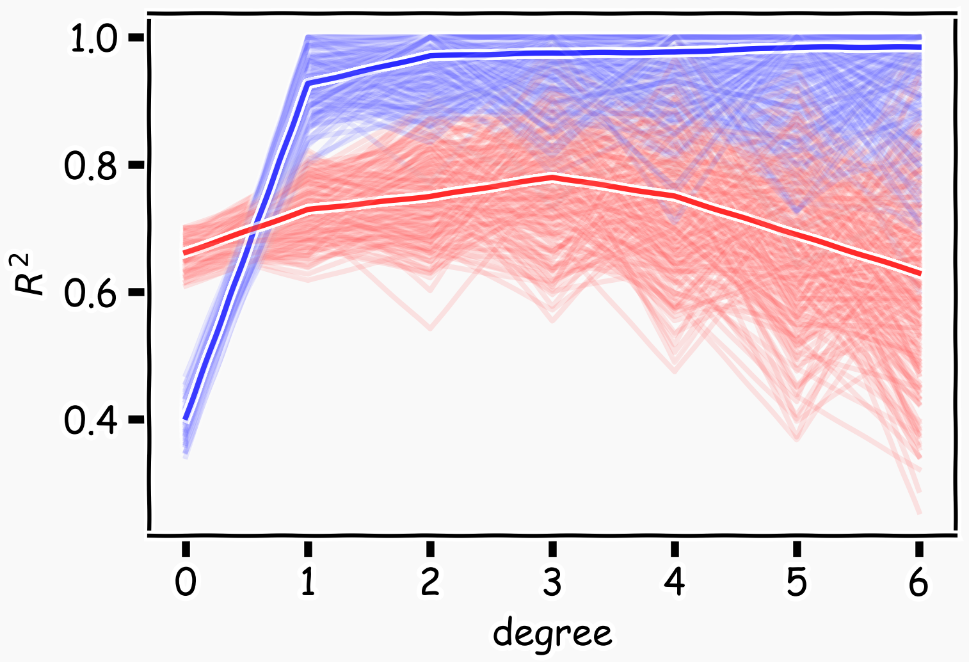
If the *k* is too small then the model is very sensitive to noise (since a new prediction is based on very few observed neighbors), and if the *k* is too large, the model tends towards making constant predictions.

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

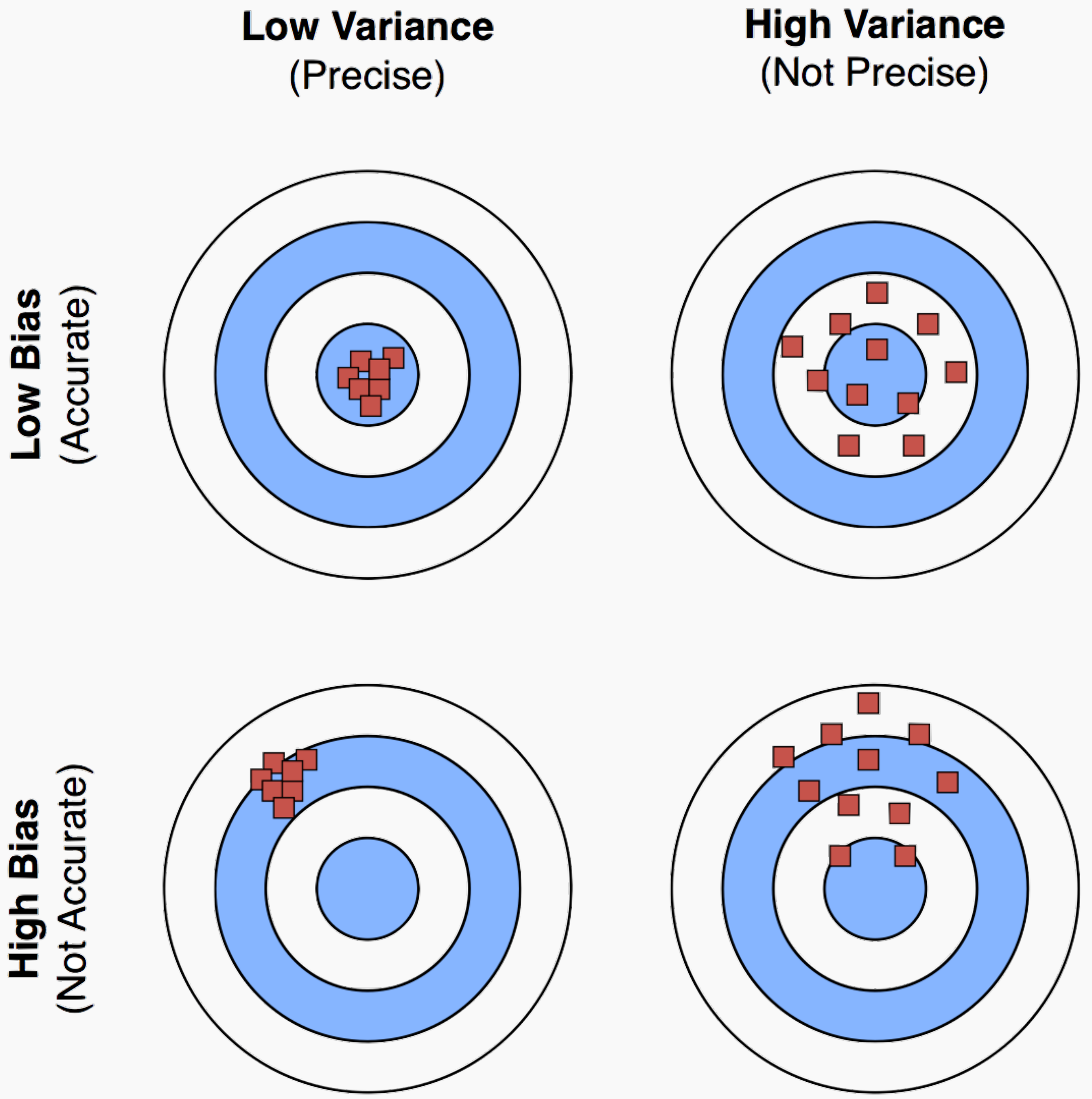
K-Fold with k=100

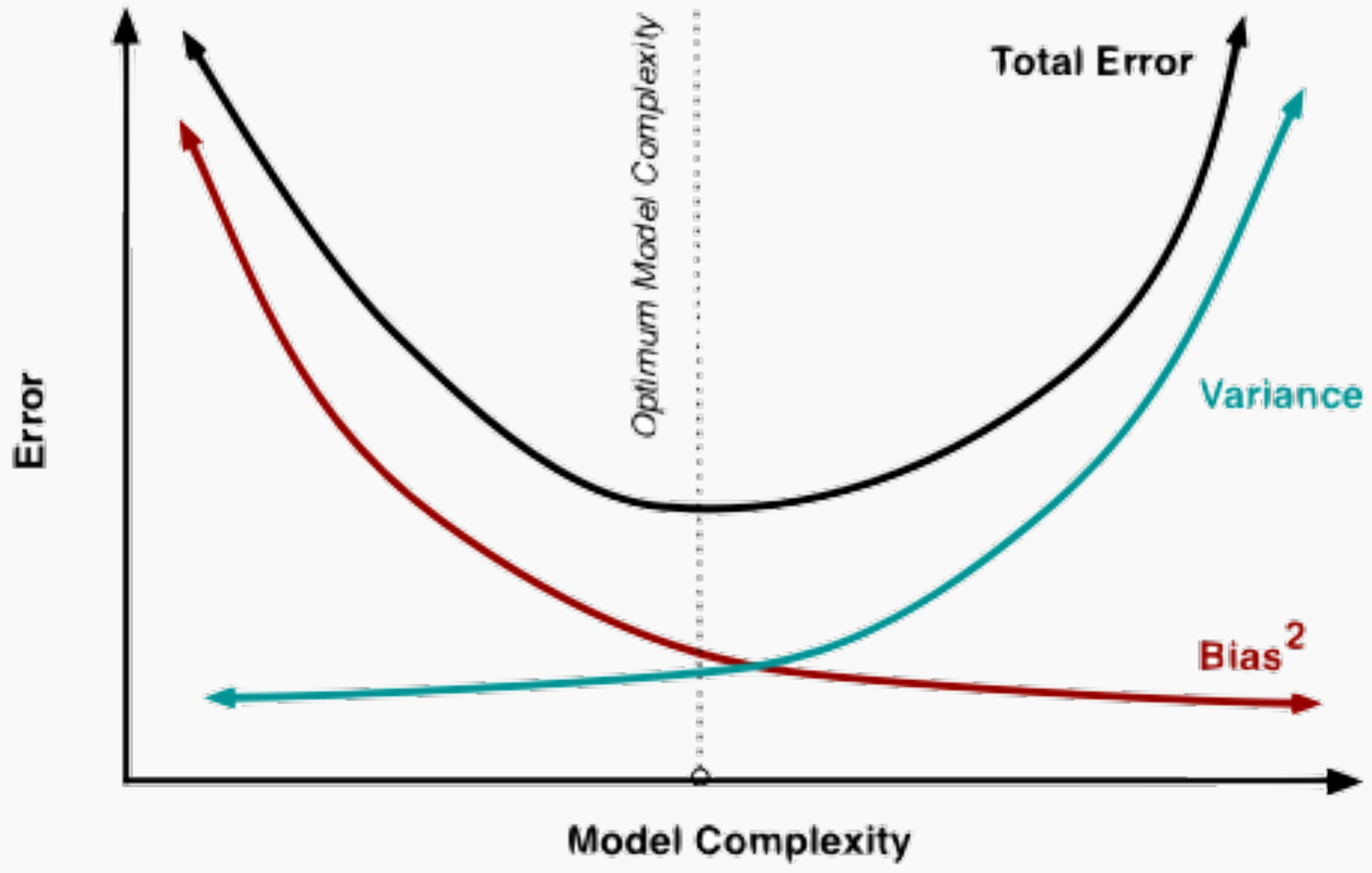


K-fold with k=100

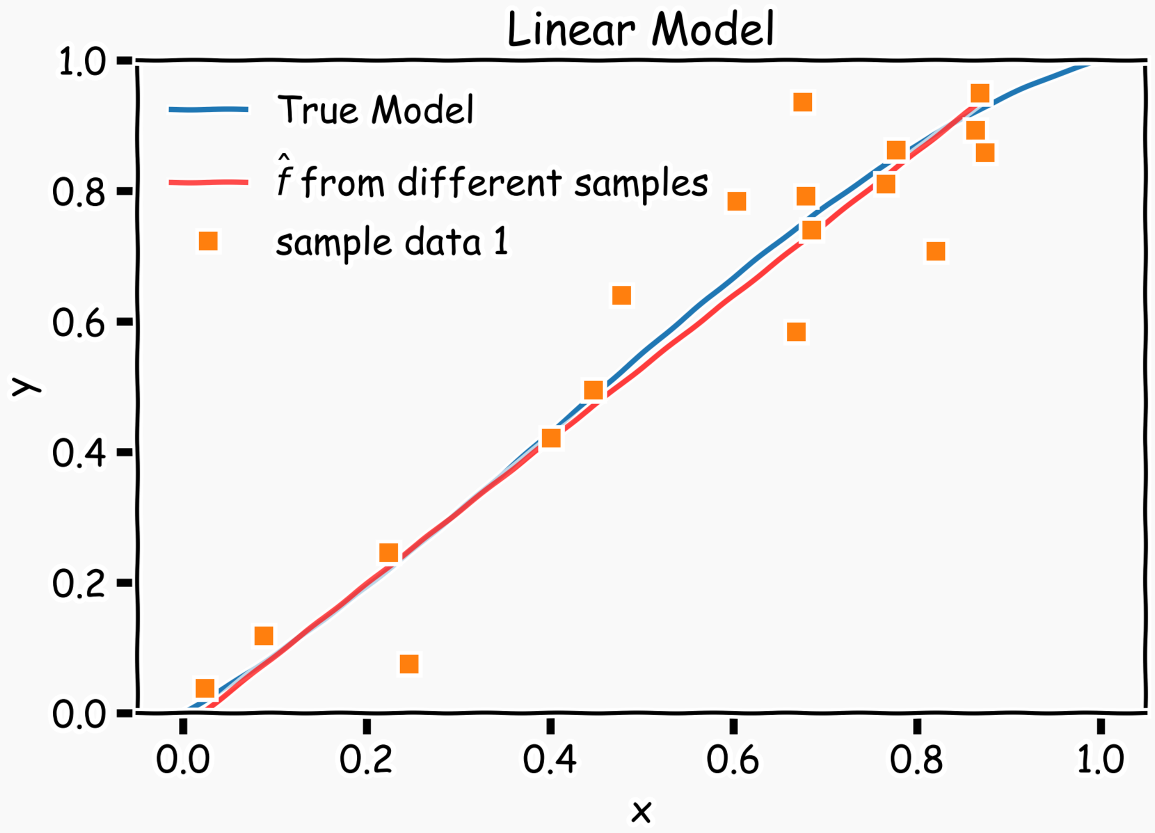


Bias vs Variance

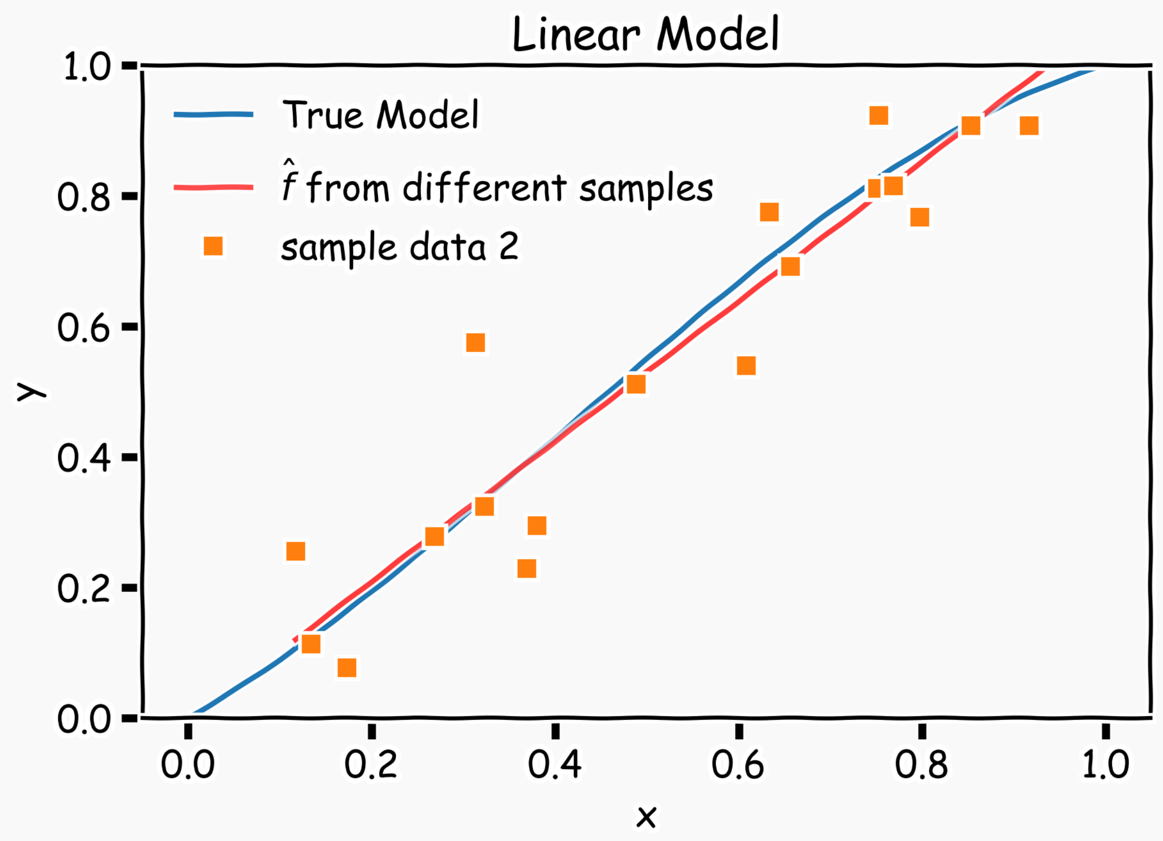

Low Variance
(Precise)
High Variance
(Not Precise)
Low Bias
(Accurate)
High Bias
(Not Accurate)


Error
Optimum Model Complexity
Total Error
Variance
Bias²
Model Complexity

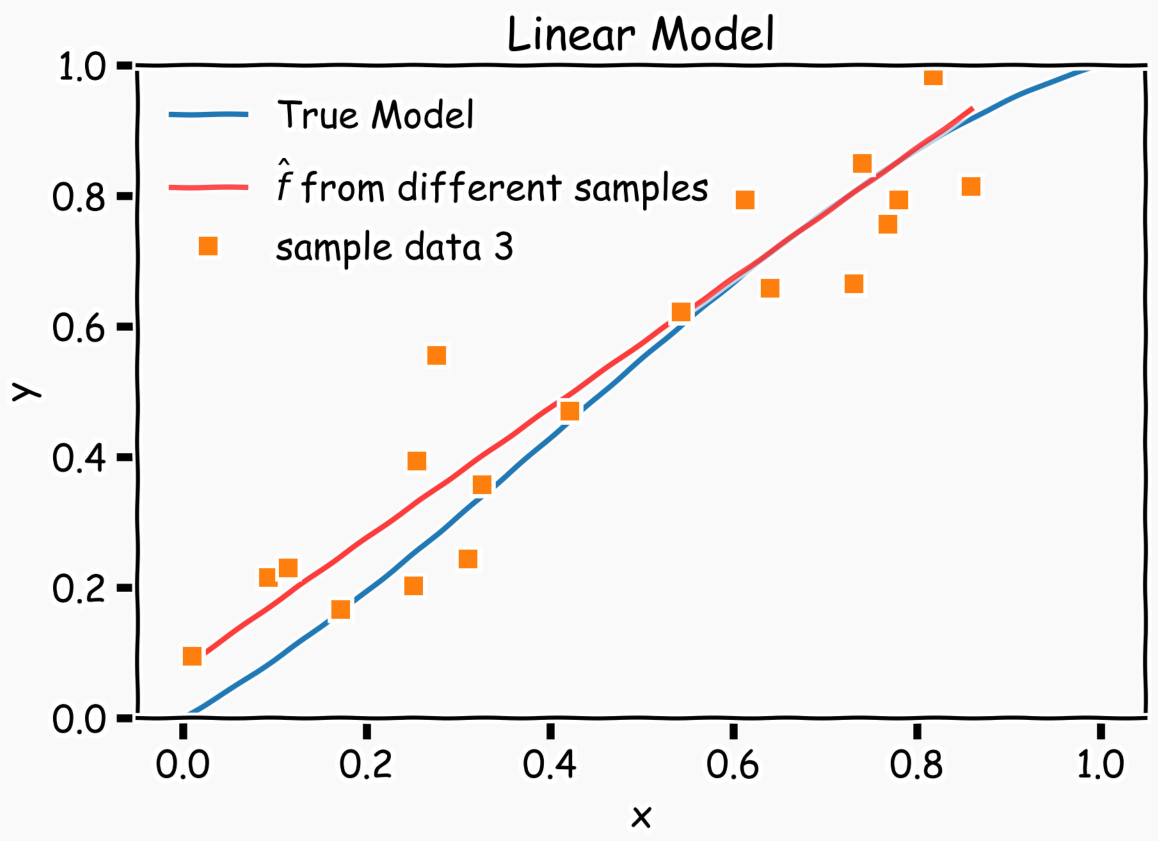
Bias vs Variance



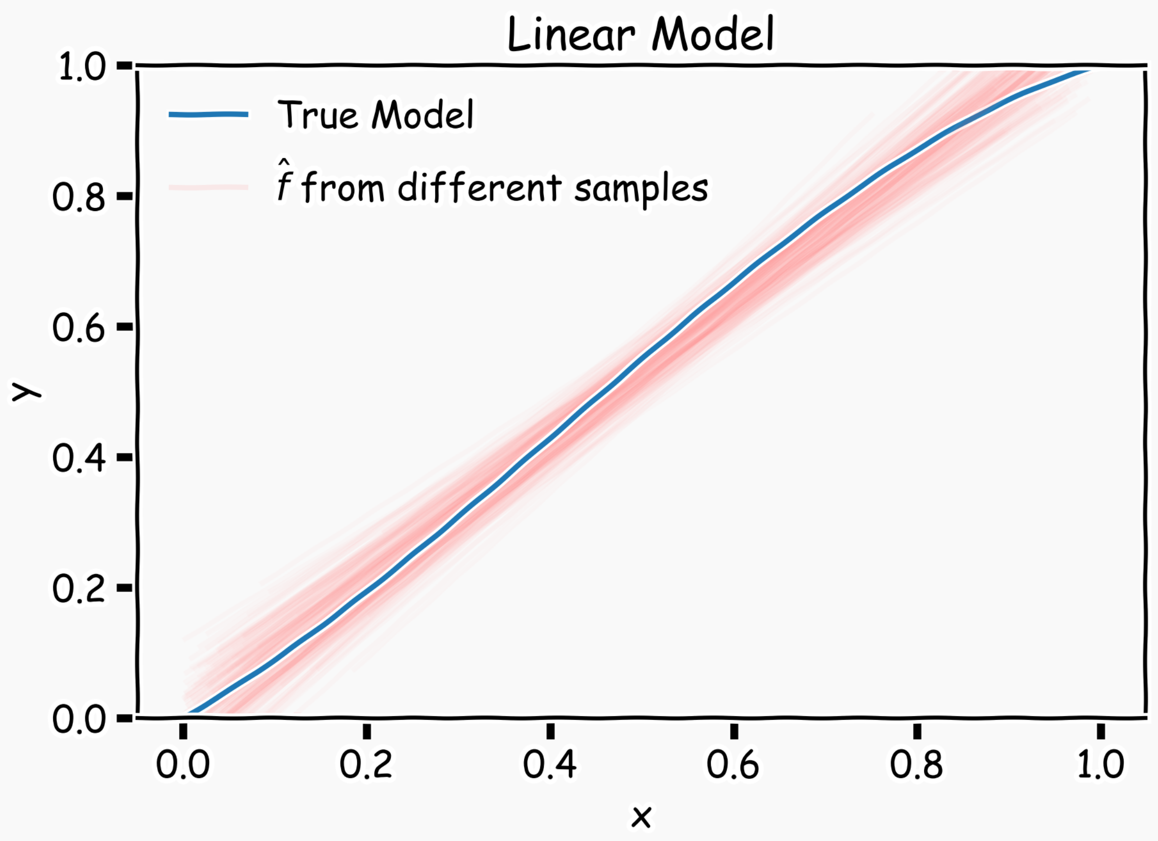
Bias vs Variance



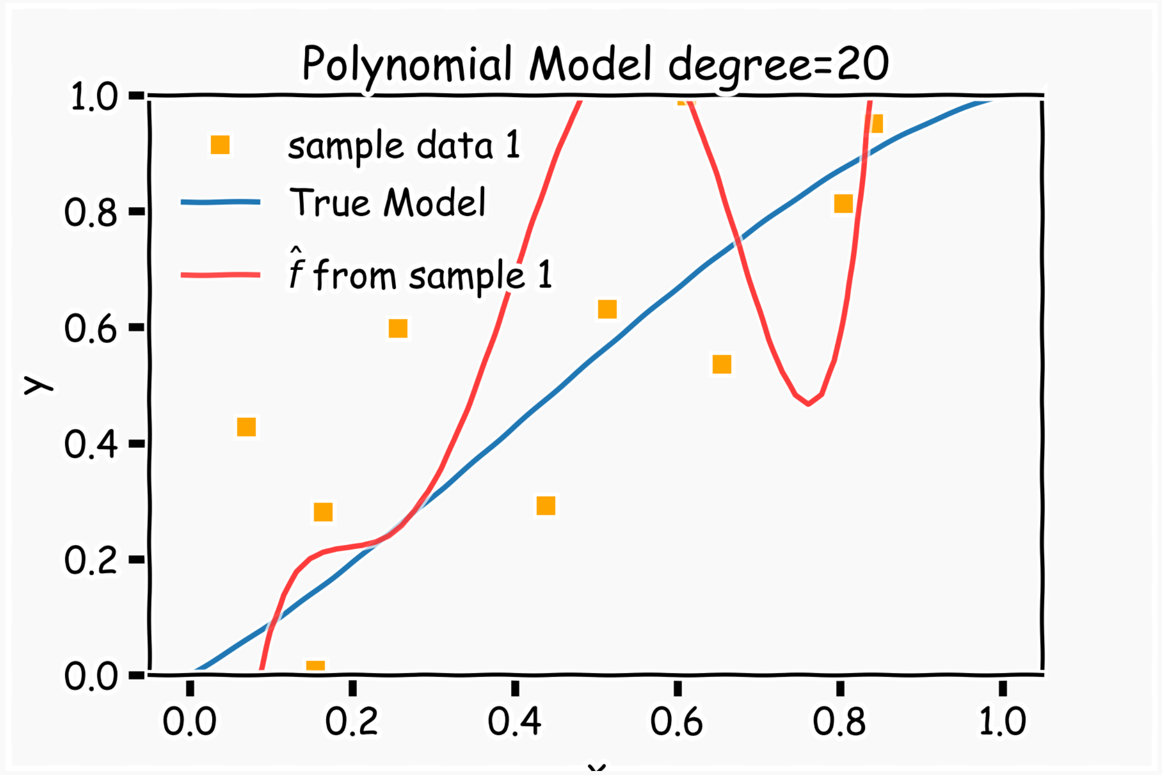
Bias vs Variance



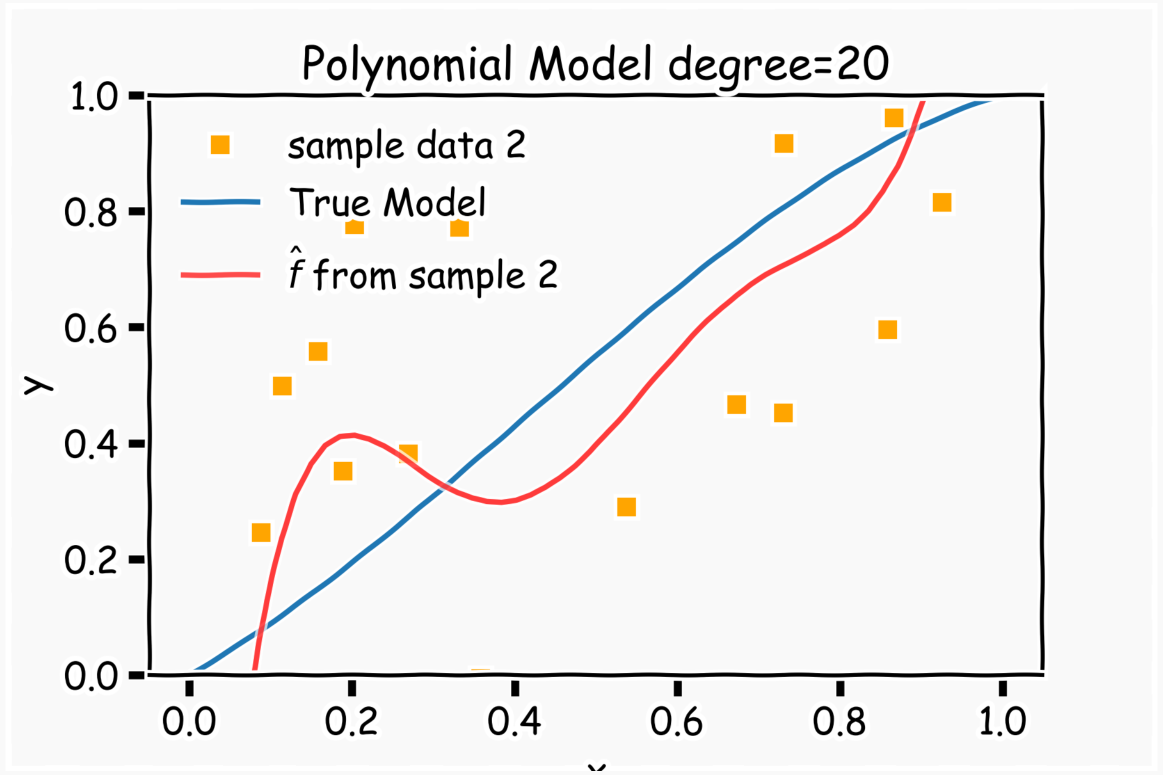
Linear models: 20 data points per line 2000 simulations



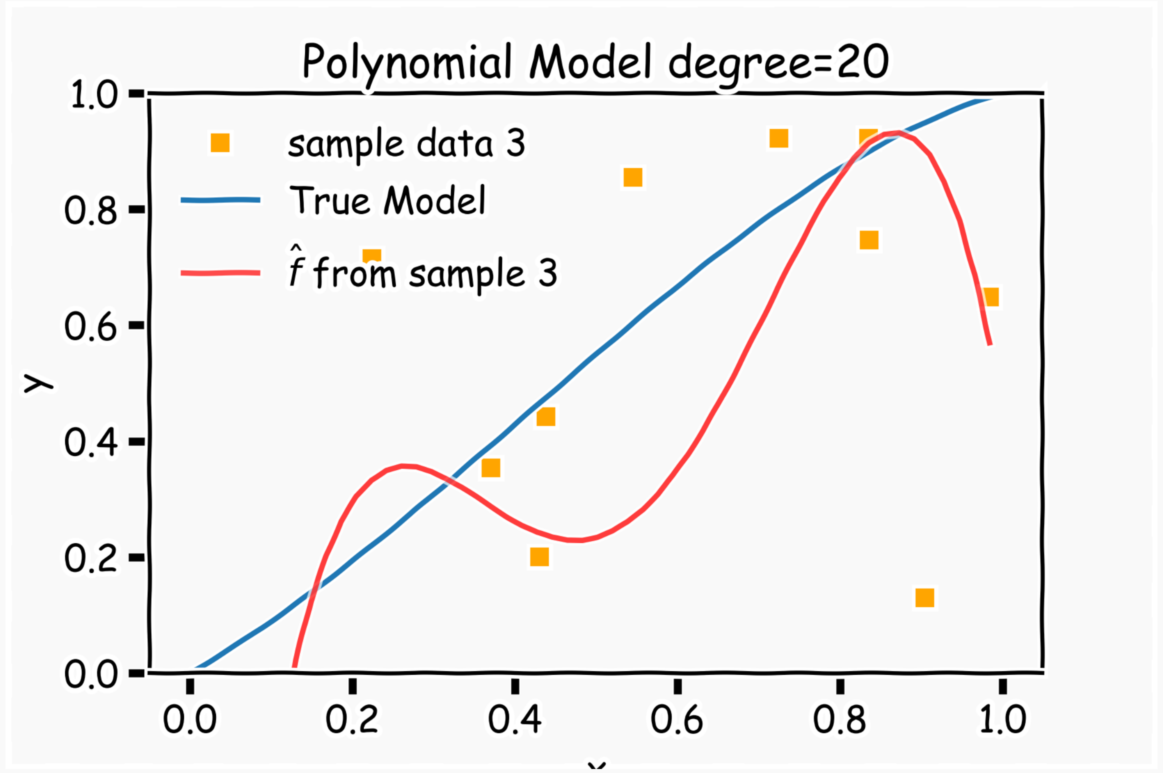
Bias vs Variance



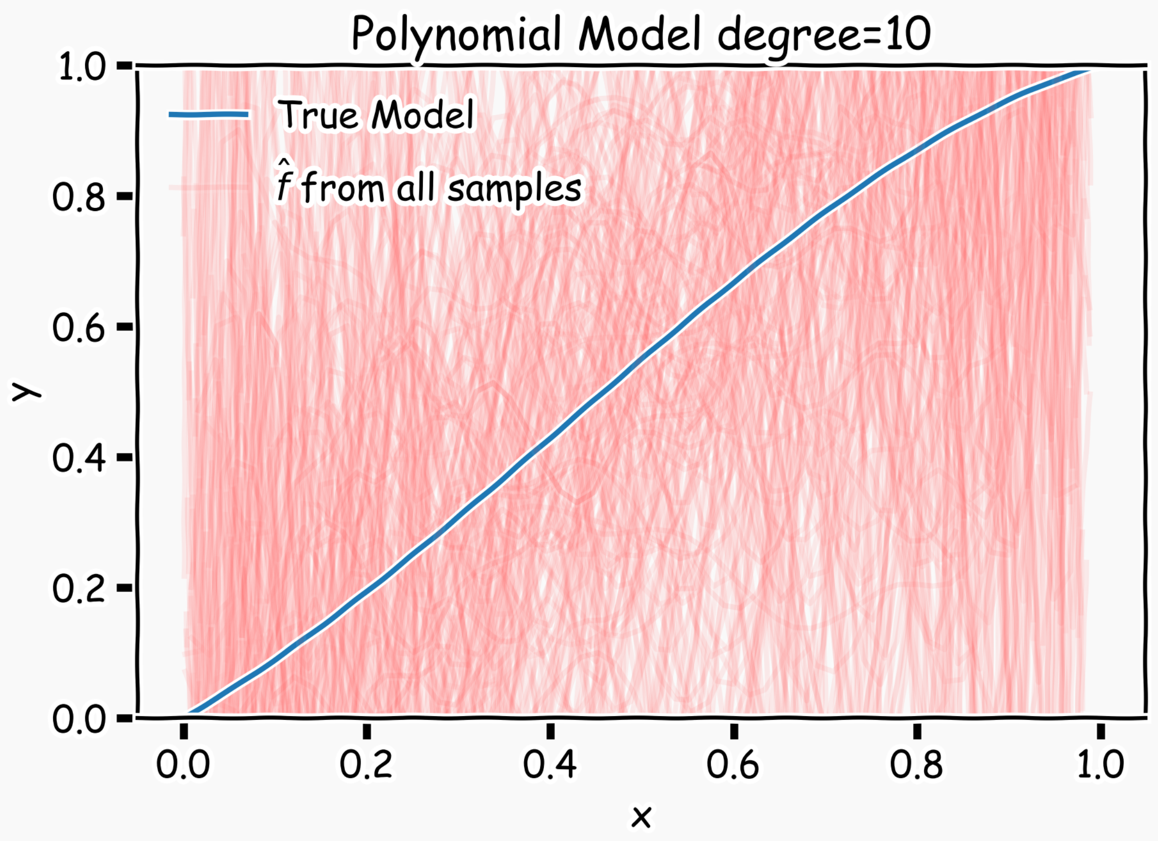
Bias vs Variance



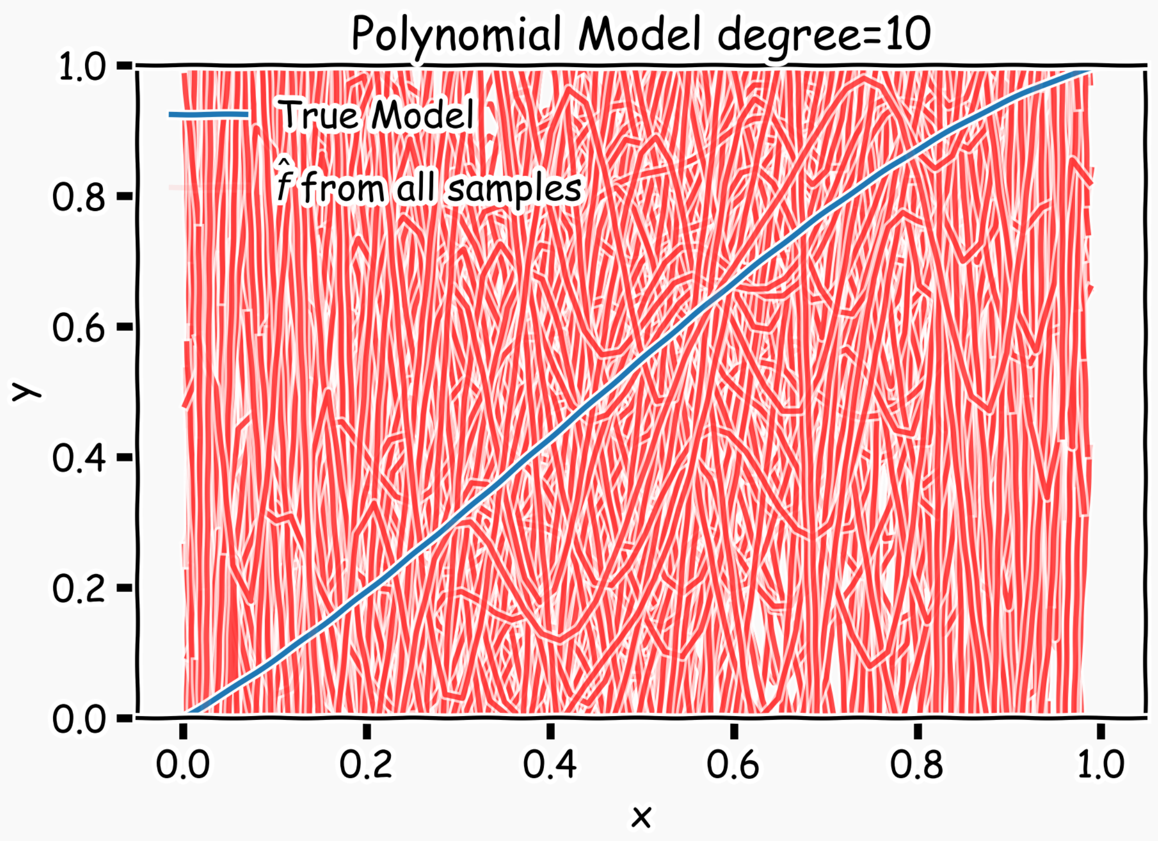
Bias vs Variance



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



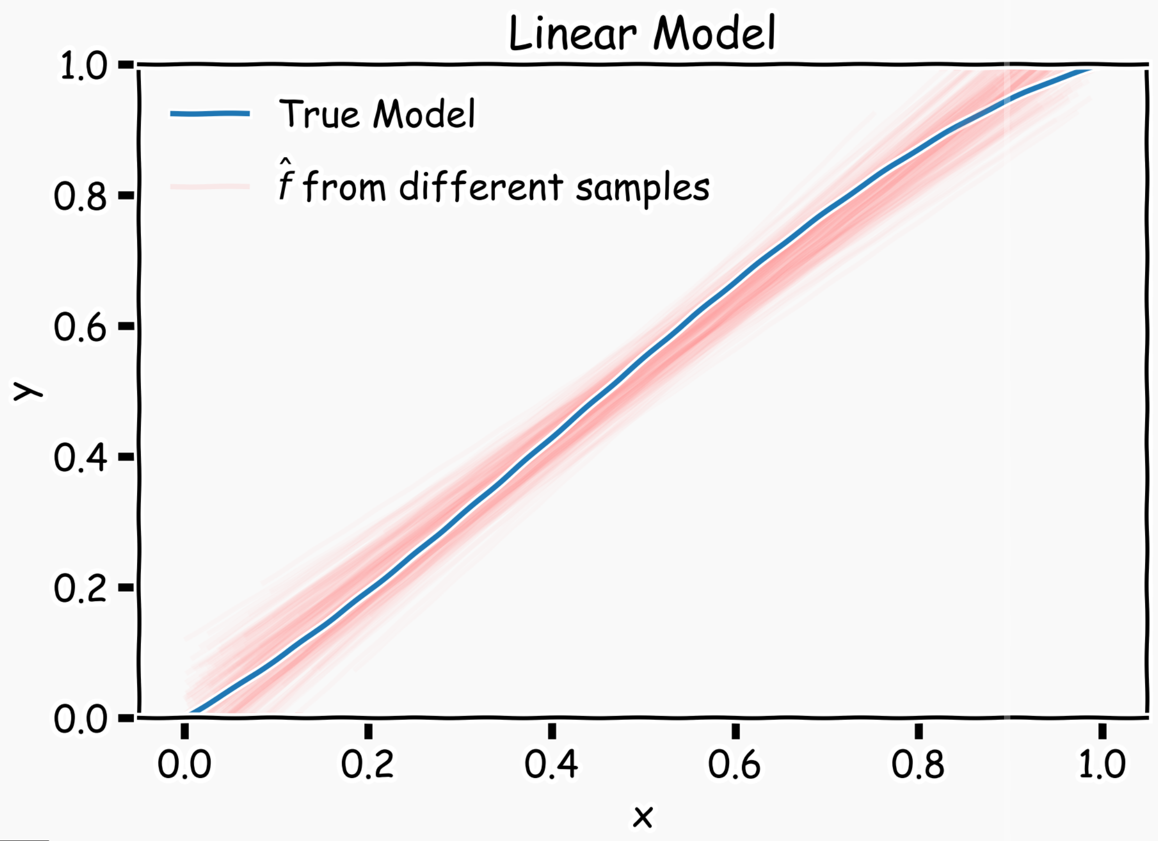
Spaghetti plot

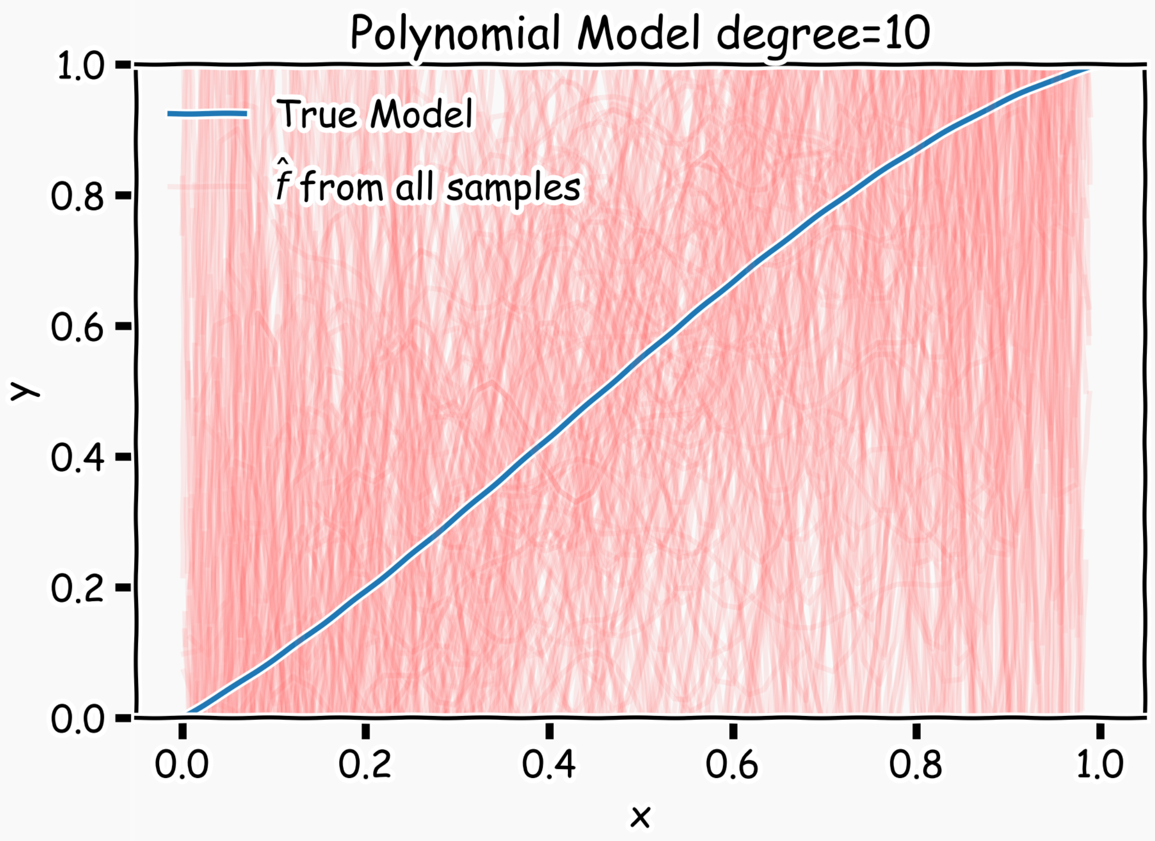


Bias vs Variance

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

Right: Best-fit models using degree 10 polynomial

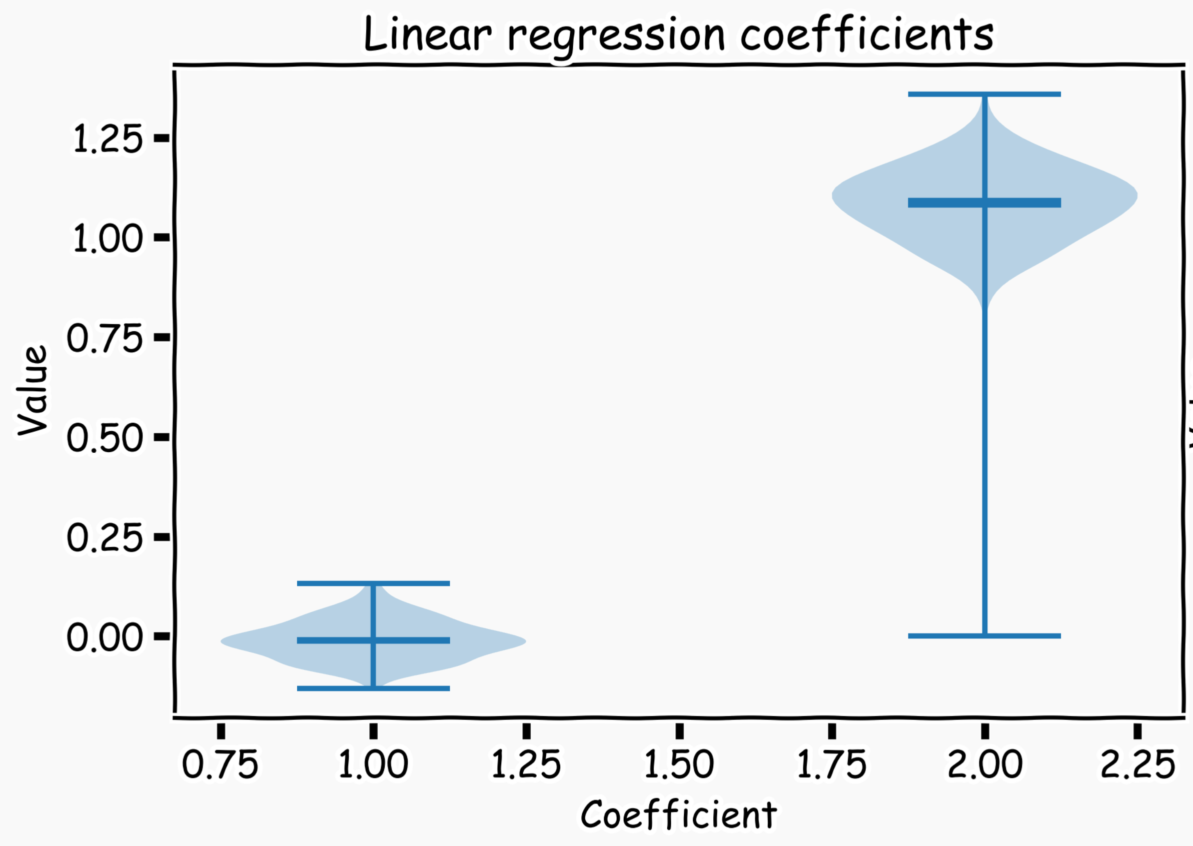


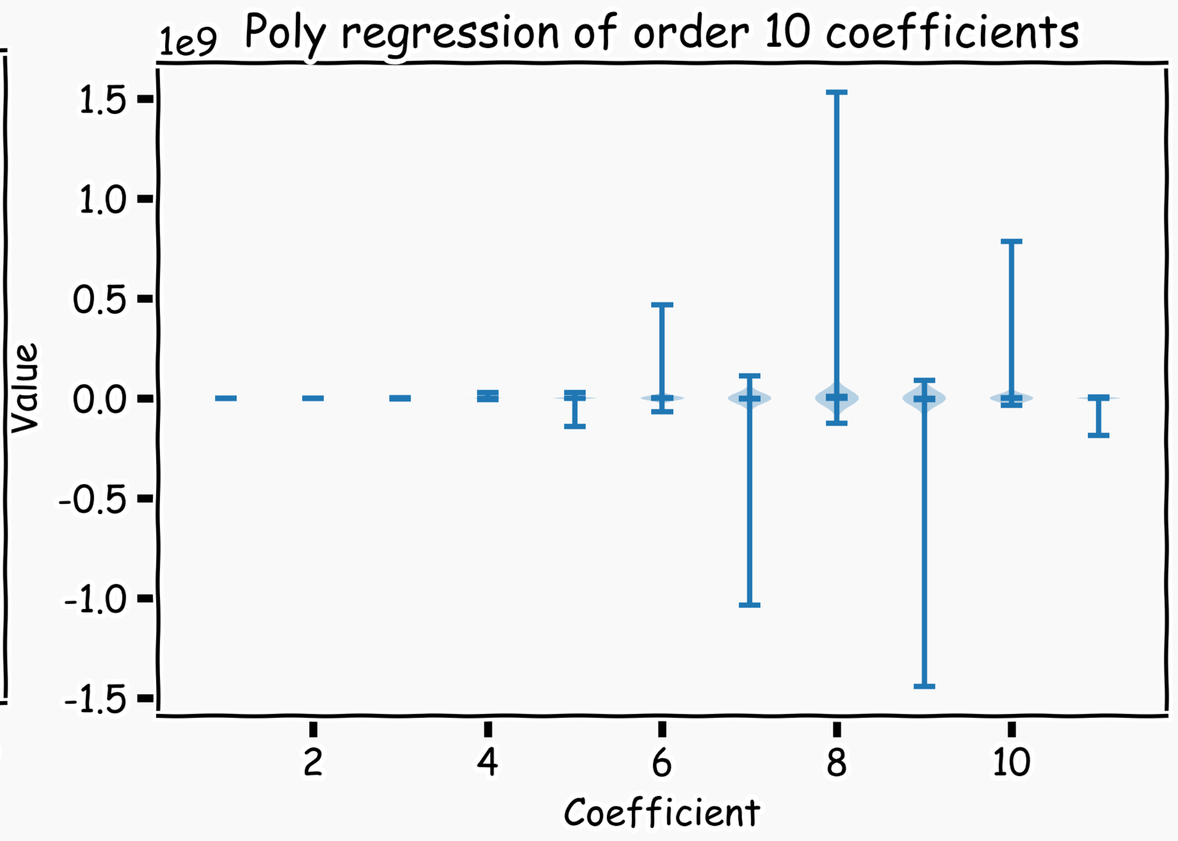


Bias vs Variance

Left: Linear regression coefficients

Right: Poly regression of order 10 coefficients





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:

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

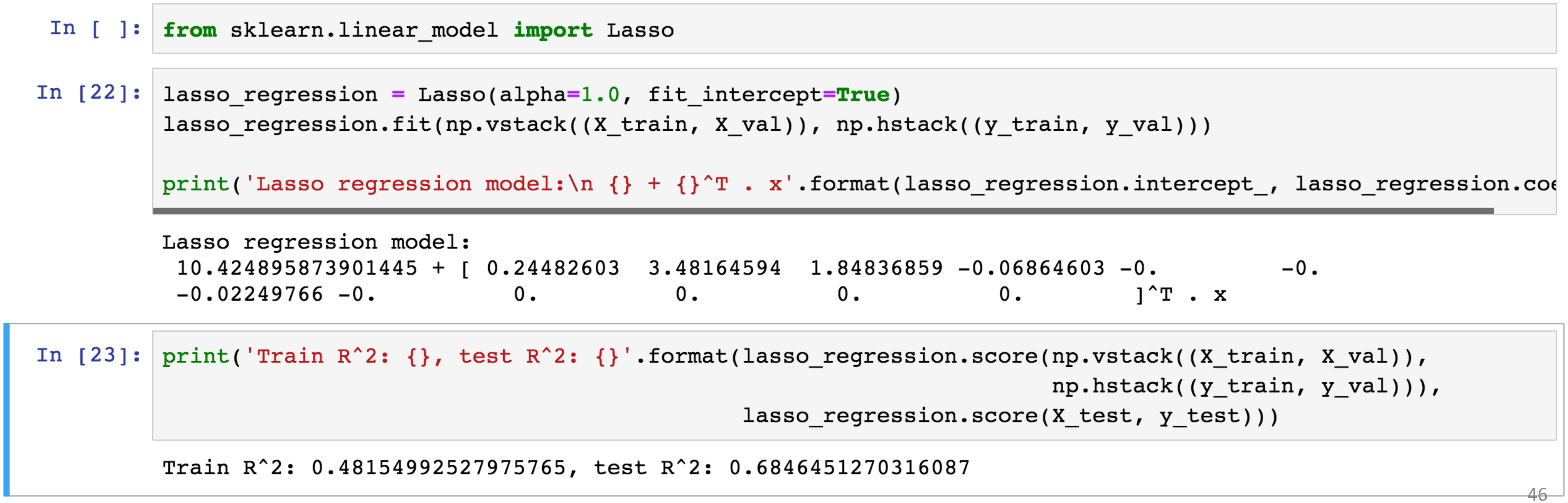
Note that $\sum_{j=1}^{J} |\beta_j|$ is the $\boldsymbol{l}_1$ norm of the vector $\boldsymbol{\beta}$

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

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.          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:

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

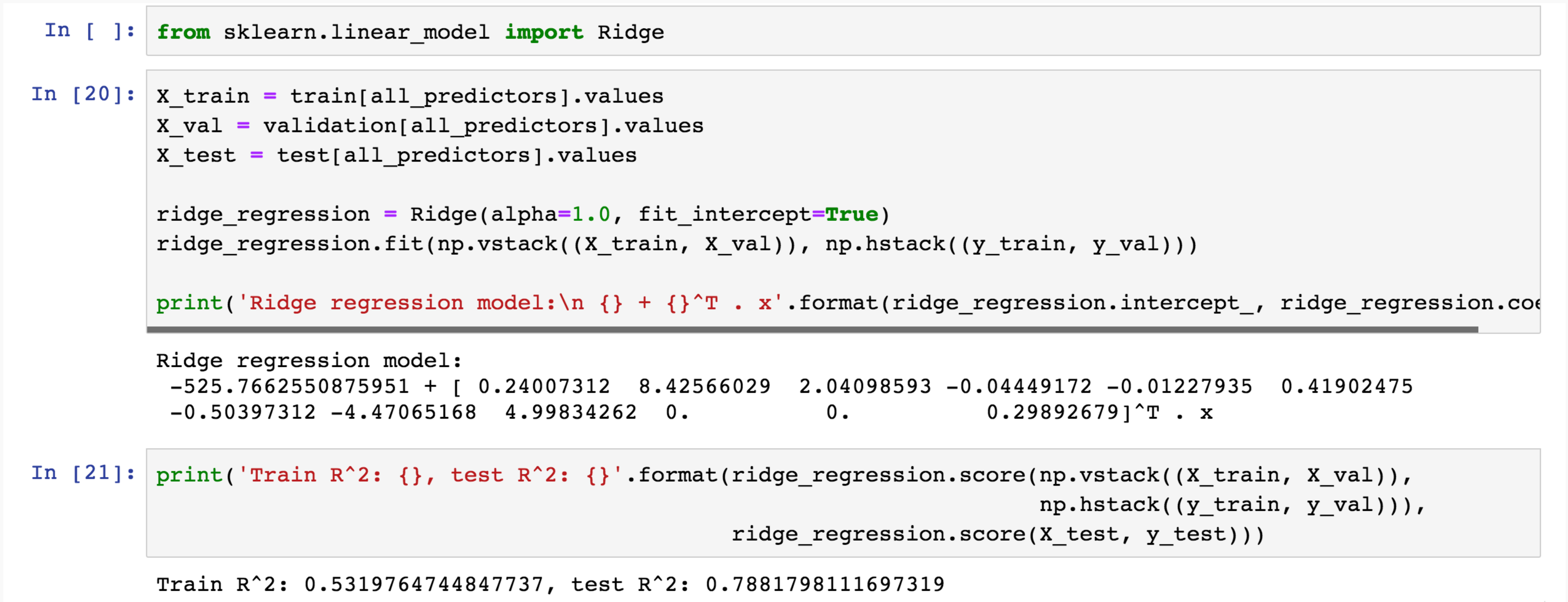
Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the l_2 norm of the vector $\boldsymbol{\beta}$

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

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

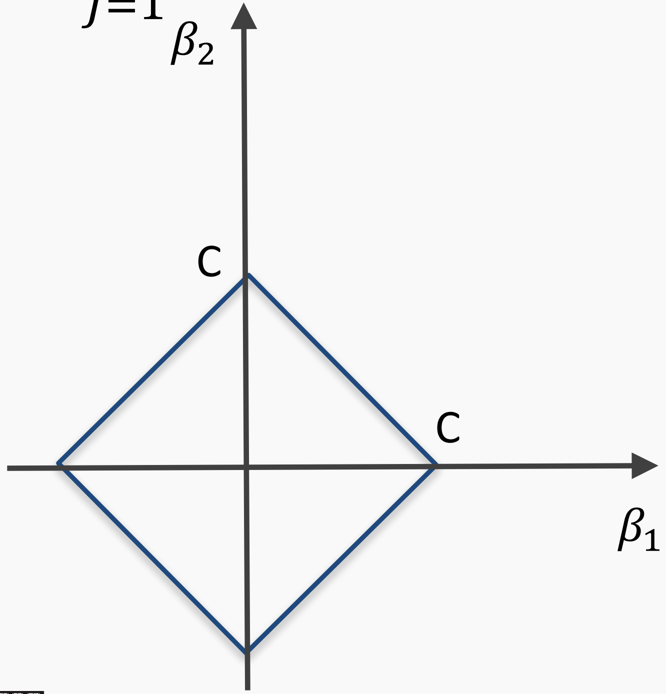
The Geometry of Regularization (LASSO)

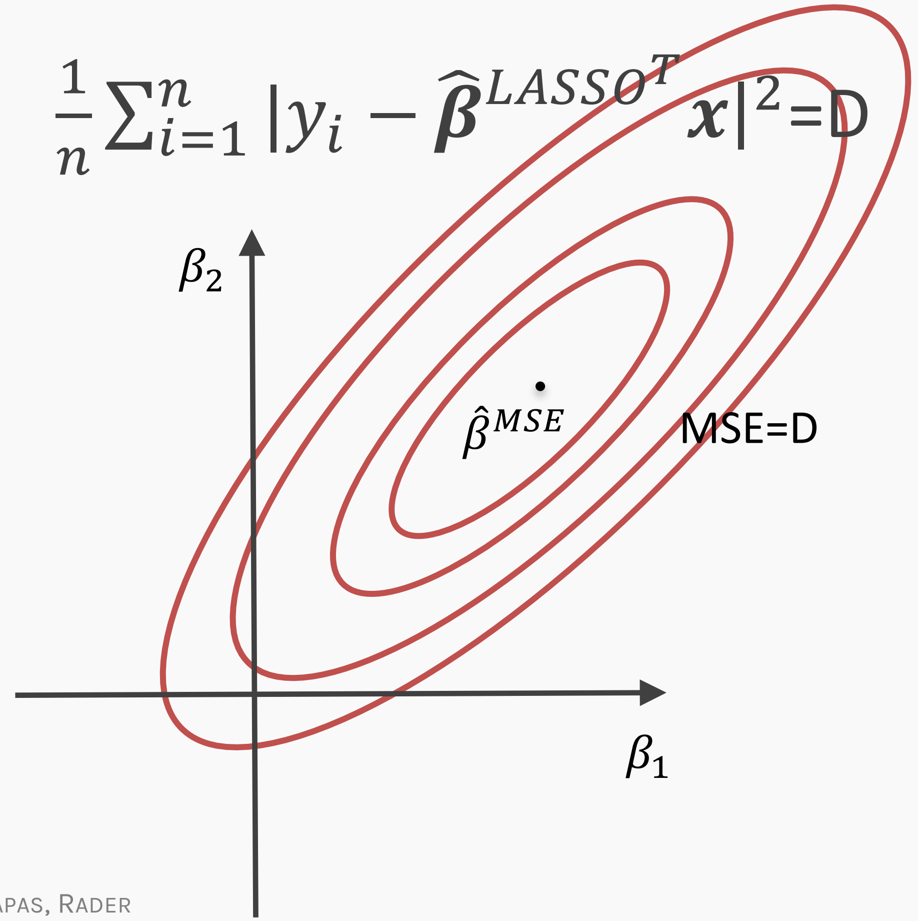
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \boldsymbol{\beta}^T \boldsymbol{x}\right|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

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

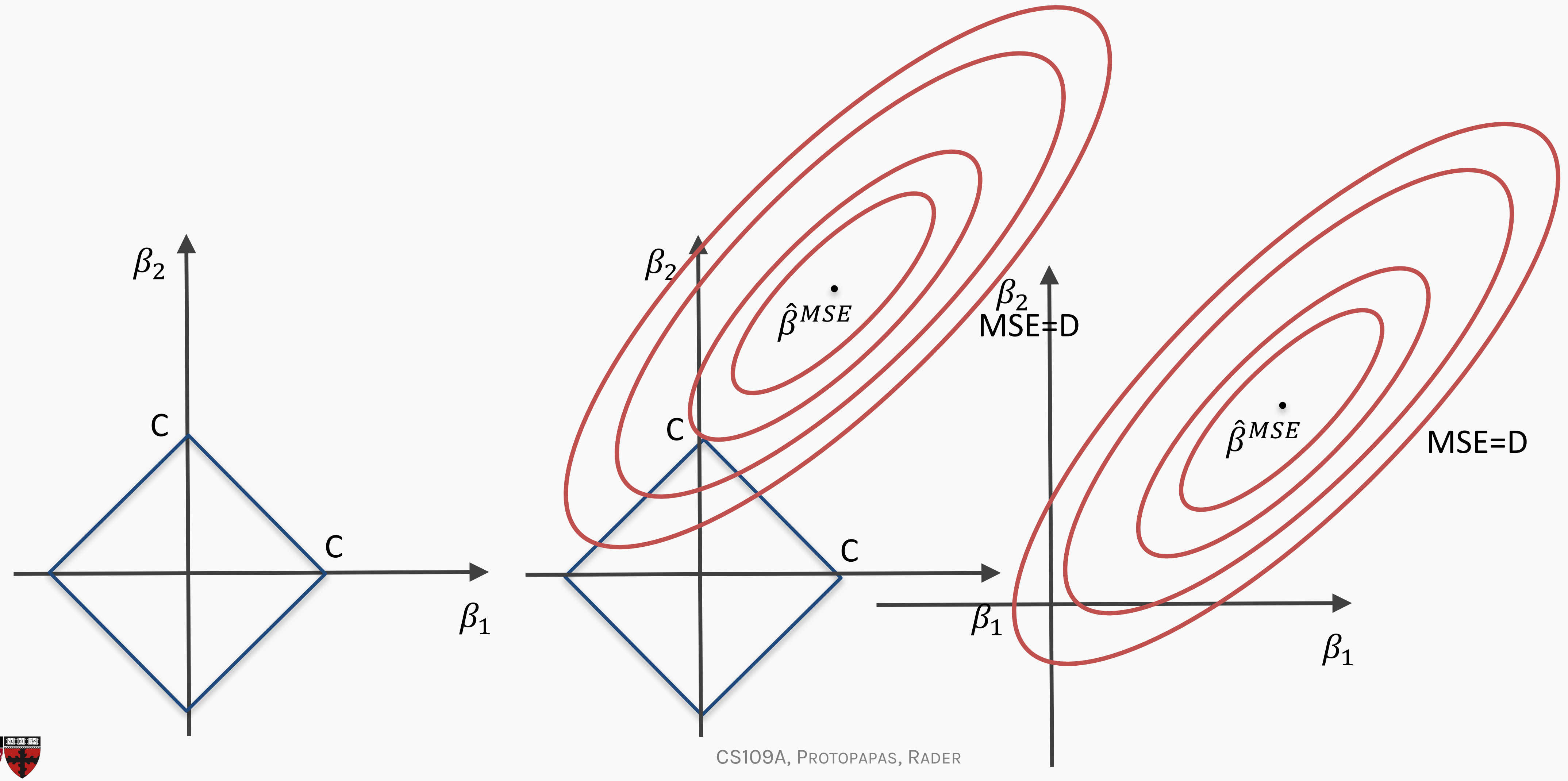
$$\lambda \sum_{j=1}^{J} \left| \hat{\beta}_j^{LASSO} \right| = C$$

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





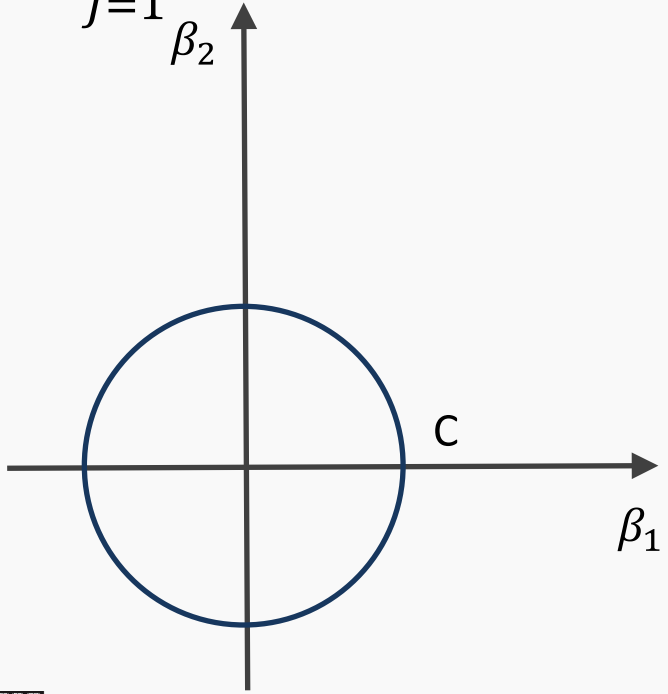
The Geometry of Regularization (LASSO)



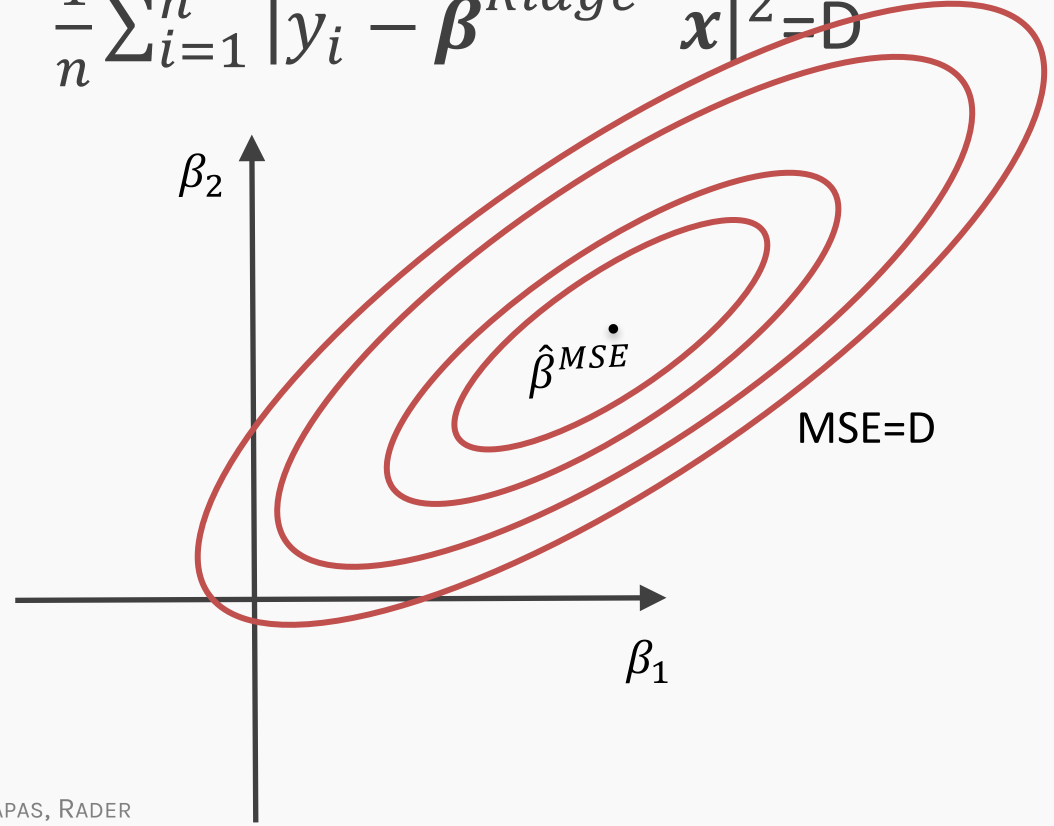
The Geometry of Regularization (Ridge)

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

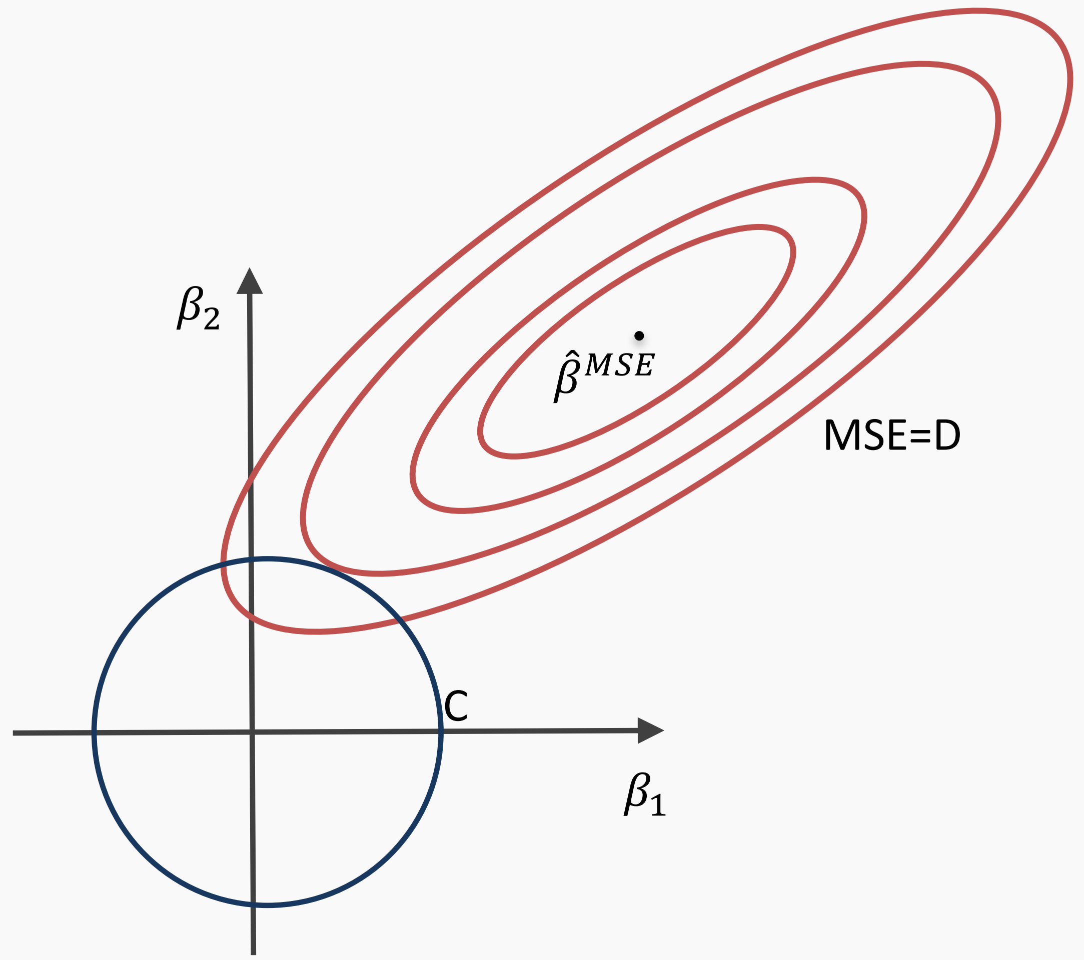
$$\lambda\sum_{j=1}^{J}\left|\hat{\beta}_j^{Ridge}\right|^2 = C$$



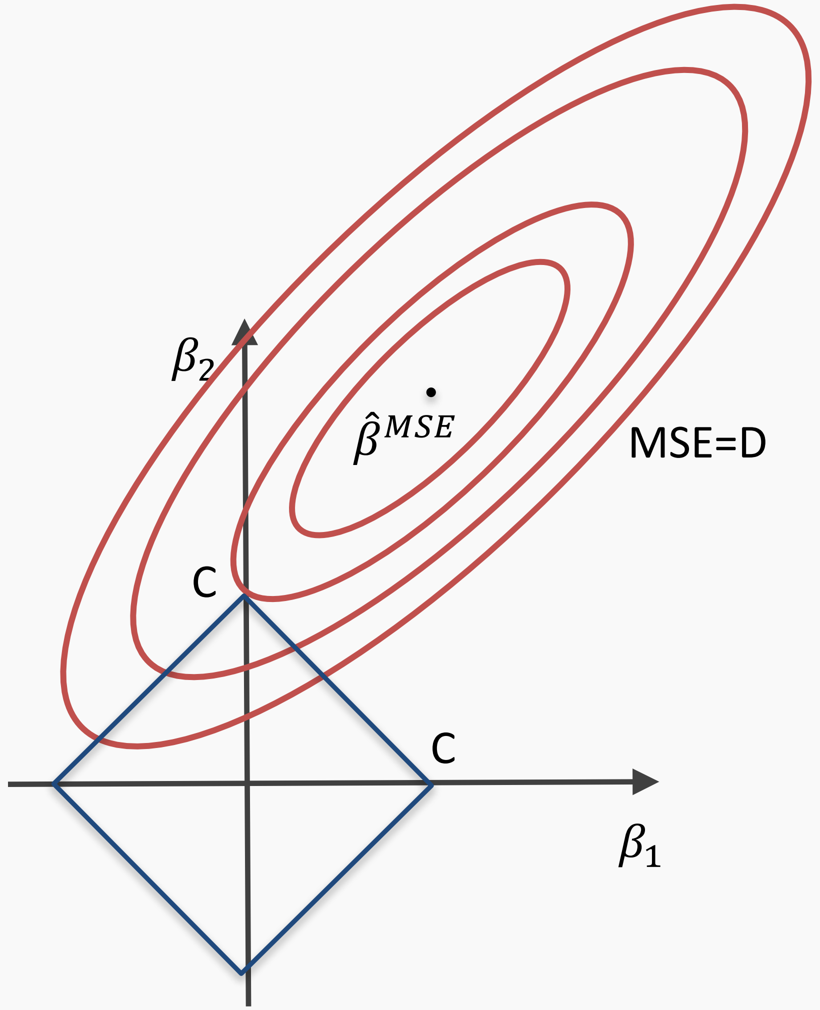
$$\frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{\boldsymbol{\beta}}^{Ridge^T}\boldsymbol{x}\right|^2 = \text{D}$$

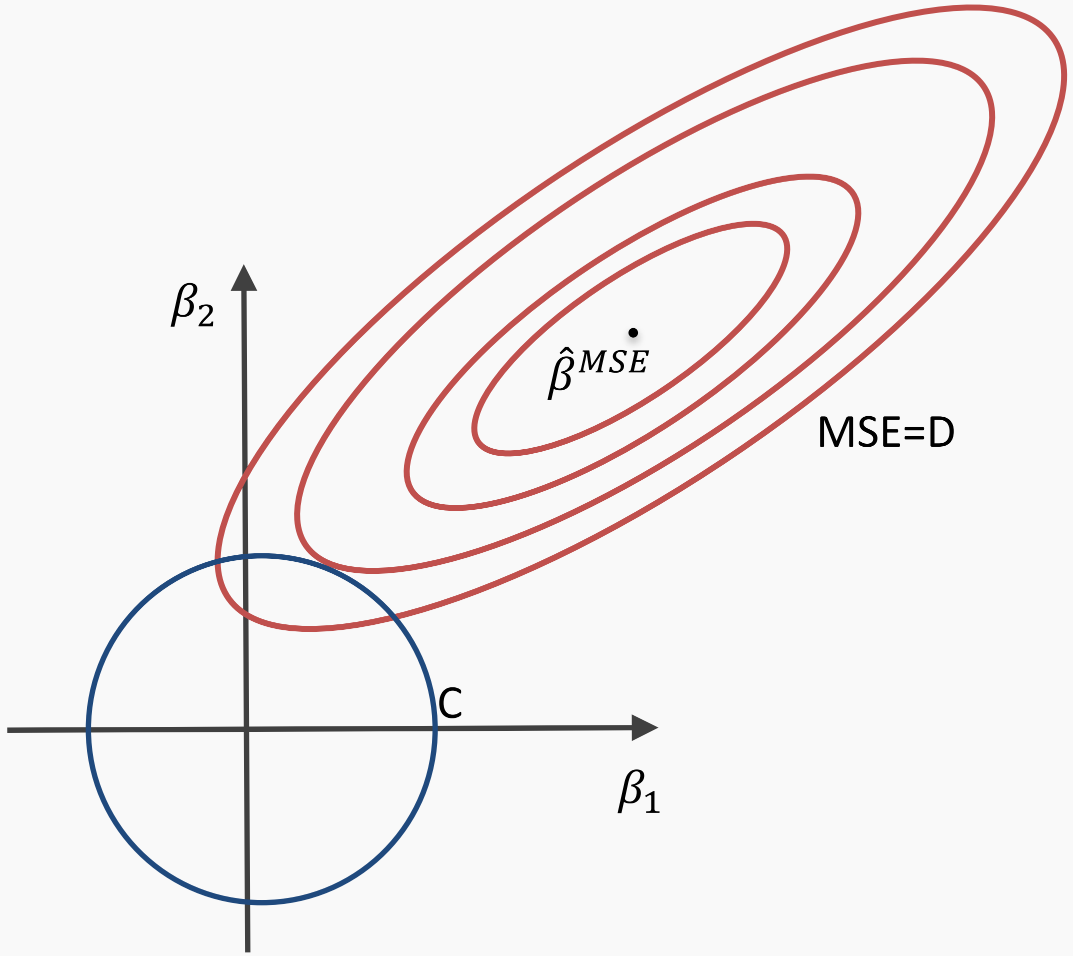


The Geometry of Regularization (Ridge)



The Geometry of Regularization





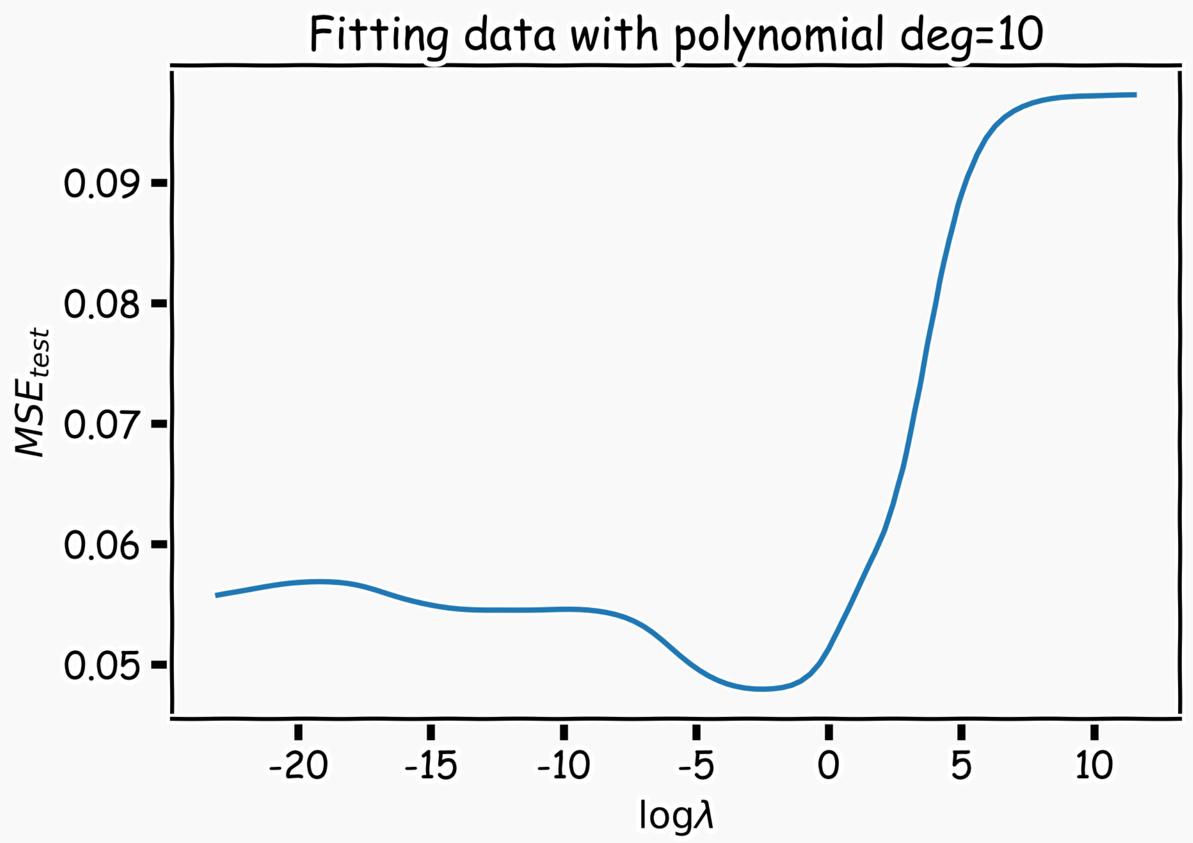
Ridge regularization with **validation** only: step by step

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \ldots \lambda_{max}\}$:
   A. 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.
   B. record $L_{MSE}(\lambda)$ using validation data.
3. select the λ that minimizes the loss on the validation data,

$$\lambda_{ridge} = \mathrm{argmin}_{\lambda}\, L_{MSE}(\lambda)$$

1. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
2. 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 $\{\lambda_{min}, \ldots \lambda_{max}\}$:
   A. determine the β that minimizes the L_{lasso}, $\hat{\beta}_{lasso}(\lambda)$, using the train data. **This is done using a solver.**

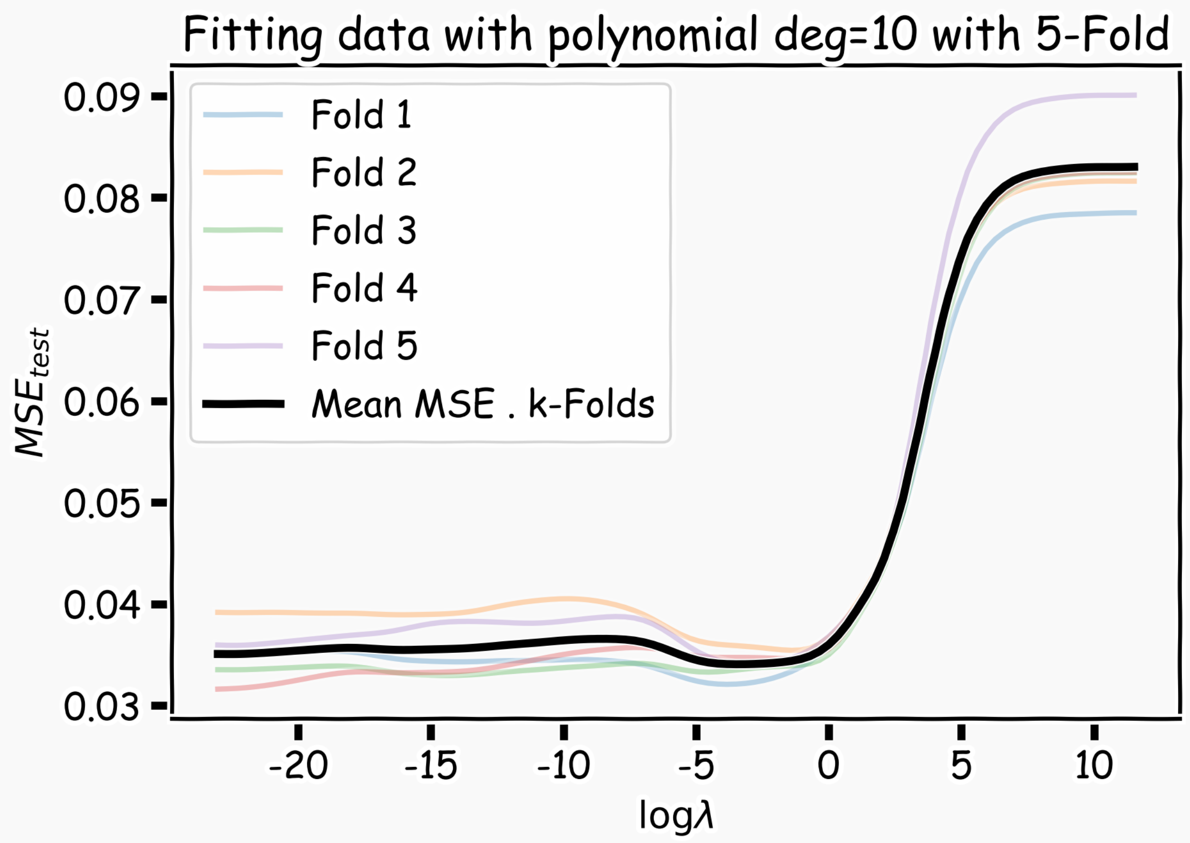
   B. record $L_{MSE}(\lambda)$ using validation data
3. select the λ that minimizes the loss on the validation data,
   $\lambda_{lasso} = \text{argmin}_{\lambda}\, L_{MSE}(\lambda)$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{lasso}(\lambda_{lasso})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{lasso}(\lambda_{lasso})$

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

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 $\hat{\beta}_{ridge}\left(\lambda_{ridge}\right)$
7. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{ridge}\left(\lambda_{ridge}\right)$

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?

Behind Ordinary Least Squares, AIC, BIC

Likelihood Functions

Recall that our statistical model for linear regression in matrix notation is:

$$Y = X\beta + \epsilon$$

It is standard to suppose that $\epsilon \sim N(0, \sigma^2)$. In fact, in many analyses we have been making this assumption. Then,

$$y | \beta, x, \epsilon \sim \mathcal{N}(x\beta, \sigma^2)$$

Question: Can you see why?

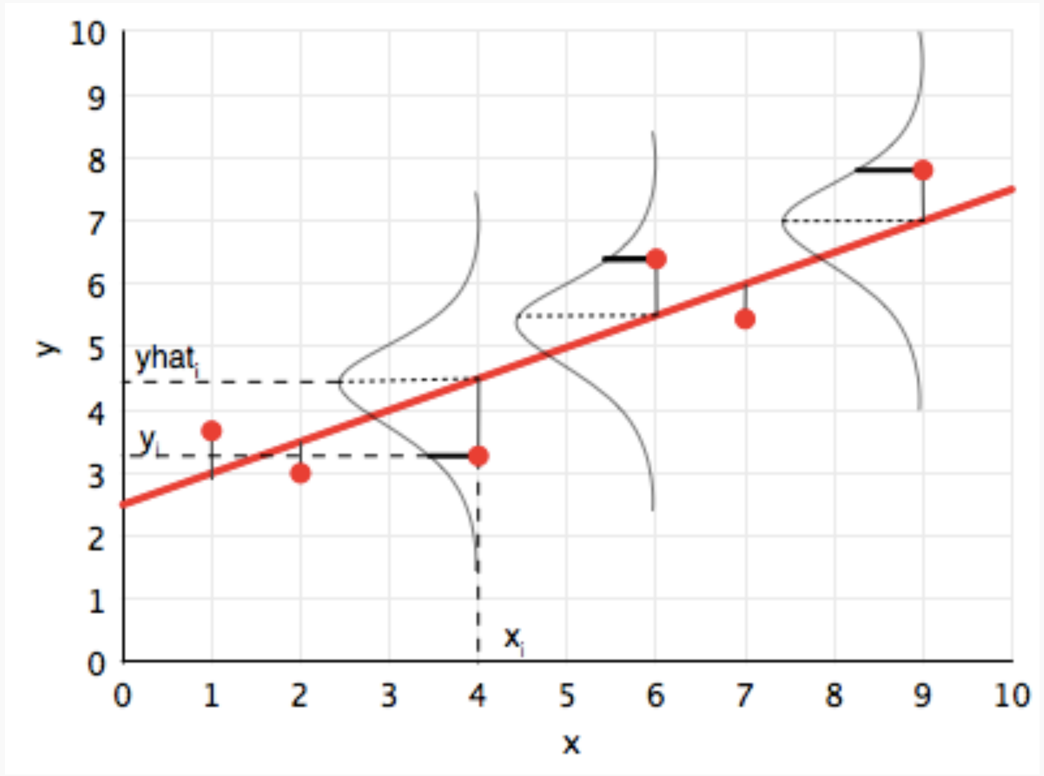
Note that $N(x\beta, \sigma^2)$ is naturally a function of the model parameters β, since the data is fixed.

Likelihood Functions

We call:

$$\mathcal{L}(\beta) = \mathcal{N}(x\beta, \sigma^2)$$

the **likelihood function**, as it gives the likelihood of the observed data for a chosen model $\boldsymbol{\beta}$.



Maximum Likelihood Estimators

Once we have a likelihood function, $\mathcal{L}(\boldsymbol{\beta})$, we have strong incentive to seek values of to maximize $\mathcal{L}$.

Can you see why?

The model parameters that maximizes $\mathcal{L}$ are called **maximum likelihood estimators (MLE)** and are denoted:

$$\boldsymbol{\beta}_{\mathrm{MLE}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, \mathcal{L}(\boldsymbol{\beta})$$

The model constructed with MLE parameters assigns the highest likelihood to the observed data.

Maximum Likelihood Estimators

But how does one maximize a likelihood function?

Fix a set of *n* observations of *J* predictors, **X**, and a set of corresponding response values, **Y**; consider a linear model $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon$.

If we assume that $\epsilon \sim N(0, \sigma^2)$ then the likelihood for each observation is

$$\mathcal{L}_i(\boldsymbol{\beta}) = \mathcal{N}(y_i; \boldsymbol{\beta}^\top \boldsymbol{x}_i, \sigma^2)$$

and the likelihood for the entire set of data is

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} \mathcal{N}(y_i; \boldsymbol{\beta}^\top \boldsymbol{x}_i, \sigma^2)$$

Maximum Likelihood Estimators

Through some algebra, we can show that maximizing $\mathcal{L}(\boldsymbol{\beta})$, is equivalent to minimizing MSE:

$$\boldsymbol{\beta}_{MLE} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\ \mathcal{L}(\boldsymbol{\beta}) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ MSE$$

Minimizing MSE or RSS is called **ordinary least squares**.