Artificial Intelligence and Machine Learning

Linear Regression
Part 2

Probabilistic Interpretation

When faced with a regression problem, why might linear regression, and specifically why might the least-squares cost function J, be a reasonable choice?

Let us assume that the target variables and the inputs are related via the equation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)},$$

where $\epsilon^{(i)}$ is an error term

Assume Gaussian Noise

Assume $\epsilon^{(i)}$ are distributed IID (independently and identically distributed) according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ^2. We can write this assumption as "$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$." I.e., the density of $\epsilon^{(i)}$ is given by

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right).$$

This implies that

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

The notation "$p(y^{(i)}|x^{(i)};\theta)$" indicates that this is the distribution of $y^{(i)}$ given $x^{(i)}$ and parameterized by θ.

Maximizing the Likelihood Function

Note that by the independence assumption on the $\epsilon^{(i)}$'s (and hence also the $y^{(i)}$'s given the $x^{(i)}$'s), this can also be written

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

Maximizing the Log Likelihood Function

Instead of maximizing $L(\theta)$, we can also maximize any strictly increasing function of $L(\theta)$. In particular, the derivations will be a bit simpler if we instead maximize the **log likelihood** $\ell(\theta)$:

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}$$

Equivalent to LMSE Minimization

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2.\end{aligned}$$

Hence, maximizing $\ell(\theta)$ gives the same answer as minimizing

$$\sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$$

which we recognize to be $J(\theta)$, our original least-squares cost function.

To Summarize

Under the previous probabilistic assumptions on the data,

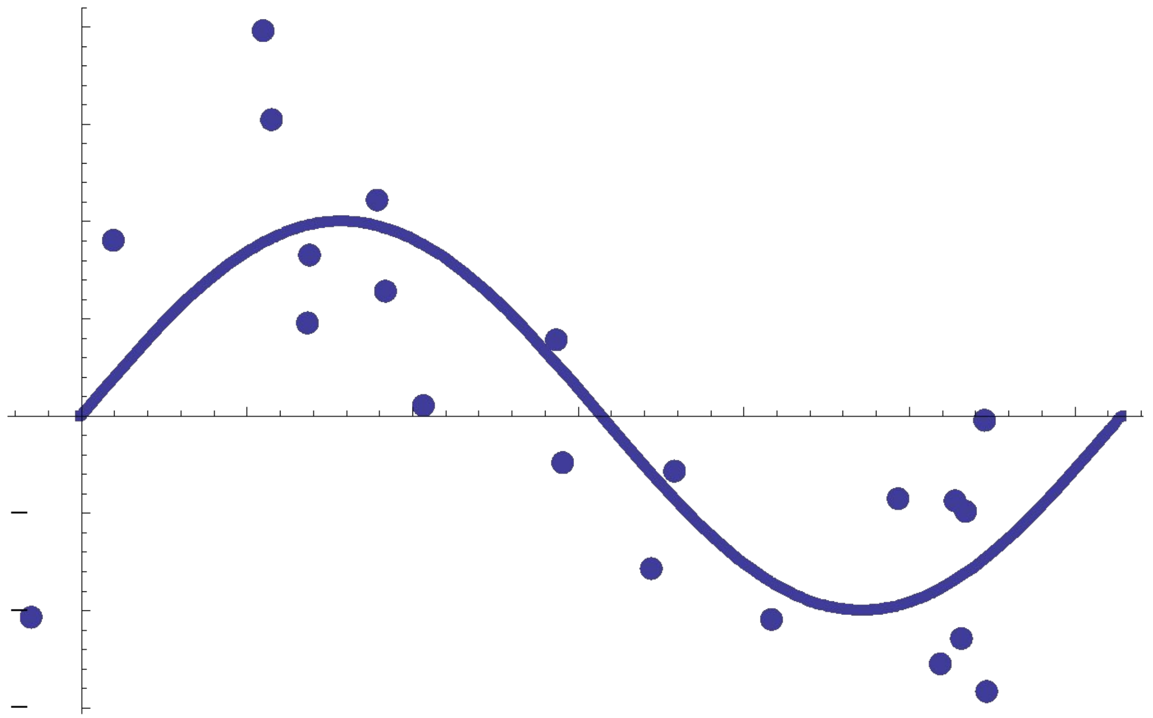
Maximizing
the Likelihood
or log Likelihood

$\Longleftrightarrow$

Minimizing
the Least-Squares
Cost/Error

Fitting Non-linear Data

- What if Y has a non-linear response?



- Can we still use a linear model?

Transforming the Feature Space

- Transform features x_i

$$x_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p})$$

- By applying non-linear transformation ϕ:

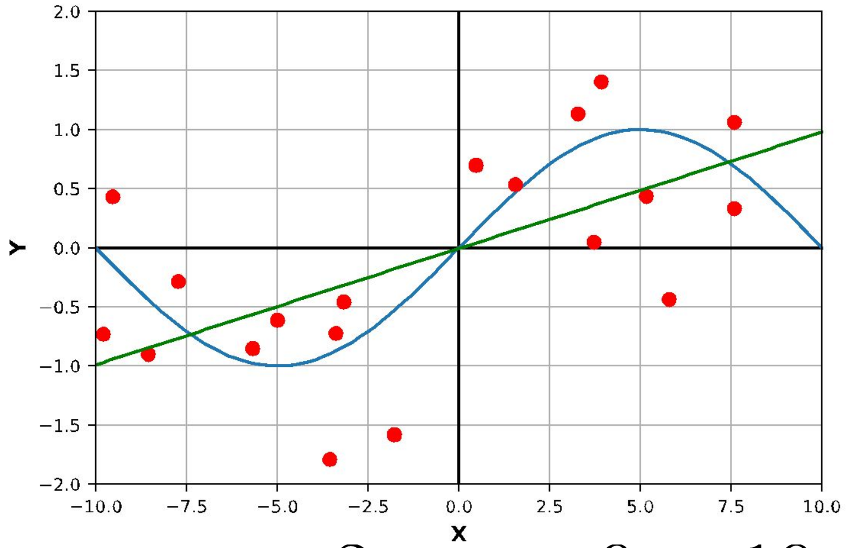
$$\phi : \mathbb{R}^p \to \mathbb{R}^k$$

- Example:

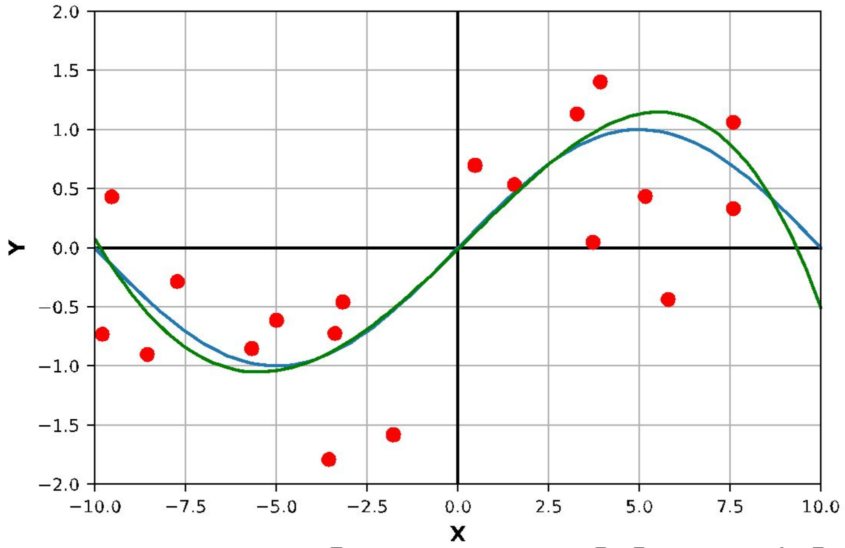
$$\phi(x) = \{1, x, x^2, \ldots, x^k\}$$

  - others: splines, radial basis functions, …
  - Expert engineered features (modeling)

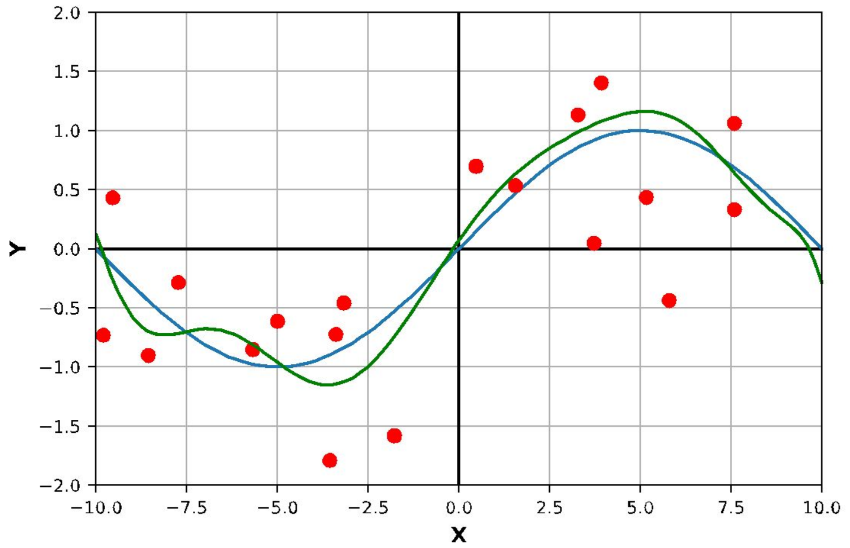
$$\{1, x\}$$



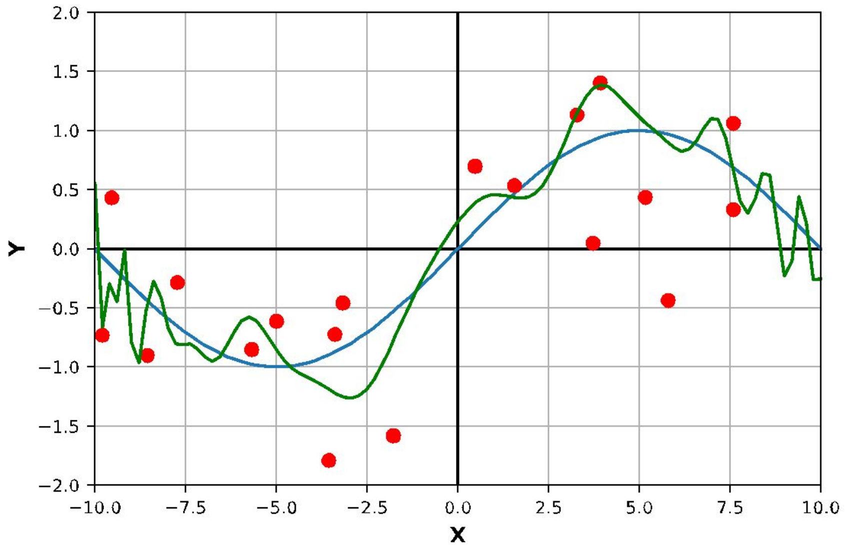
$$\{1, x, x^2, x^3, x^4\}$$



$$\{1, x, x^2, ..., x^9, x^{10}\}$$

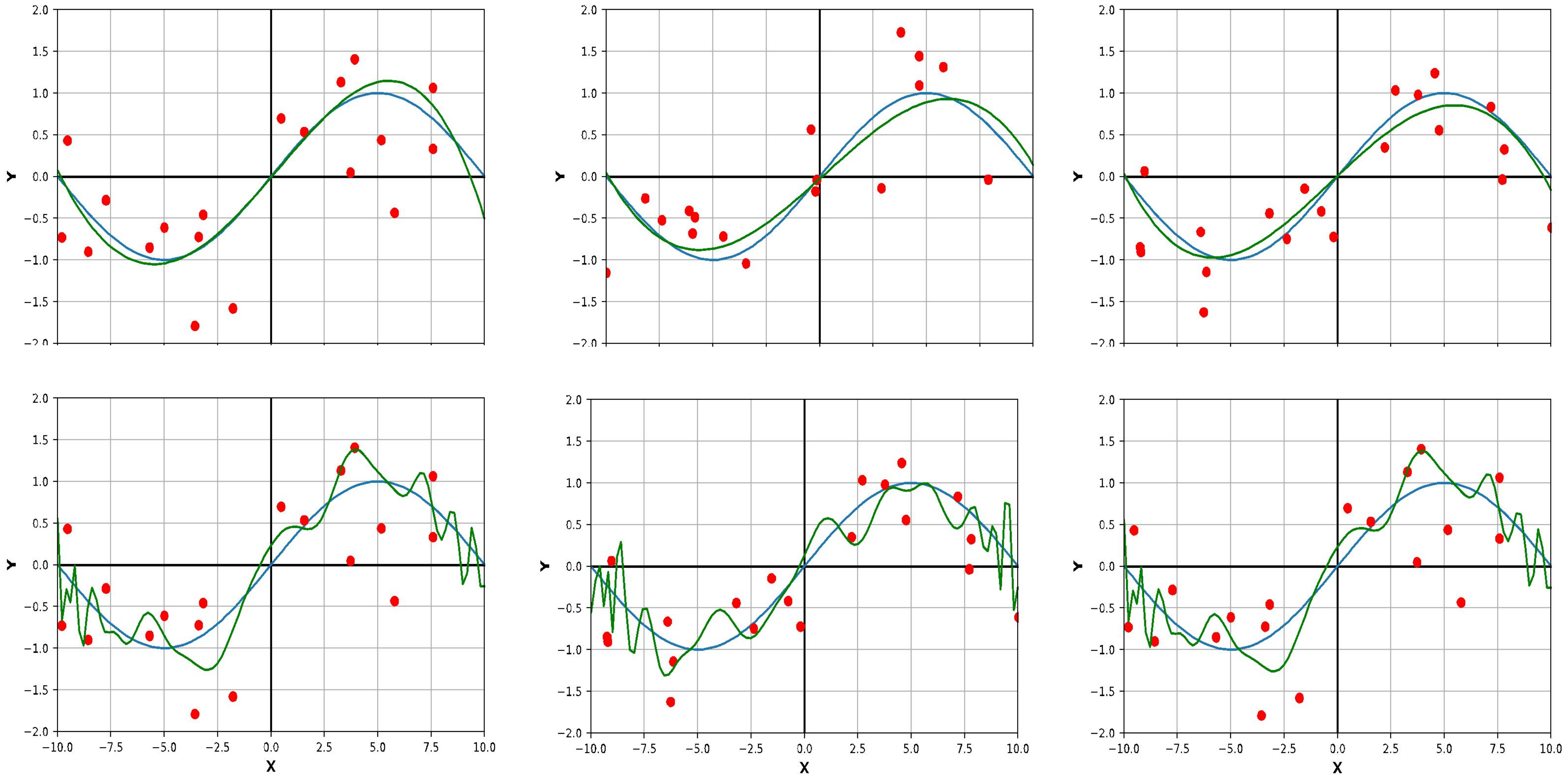


$$\{1, x, x^2, ..., x^{99}, x^{100}\}$$

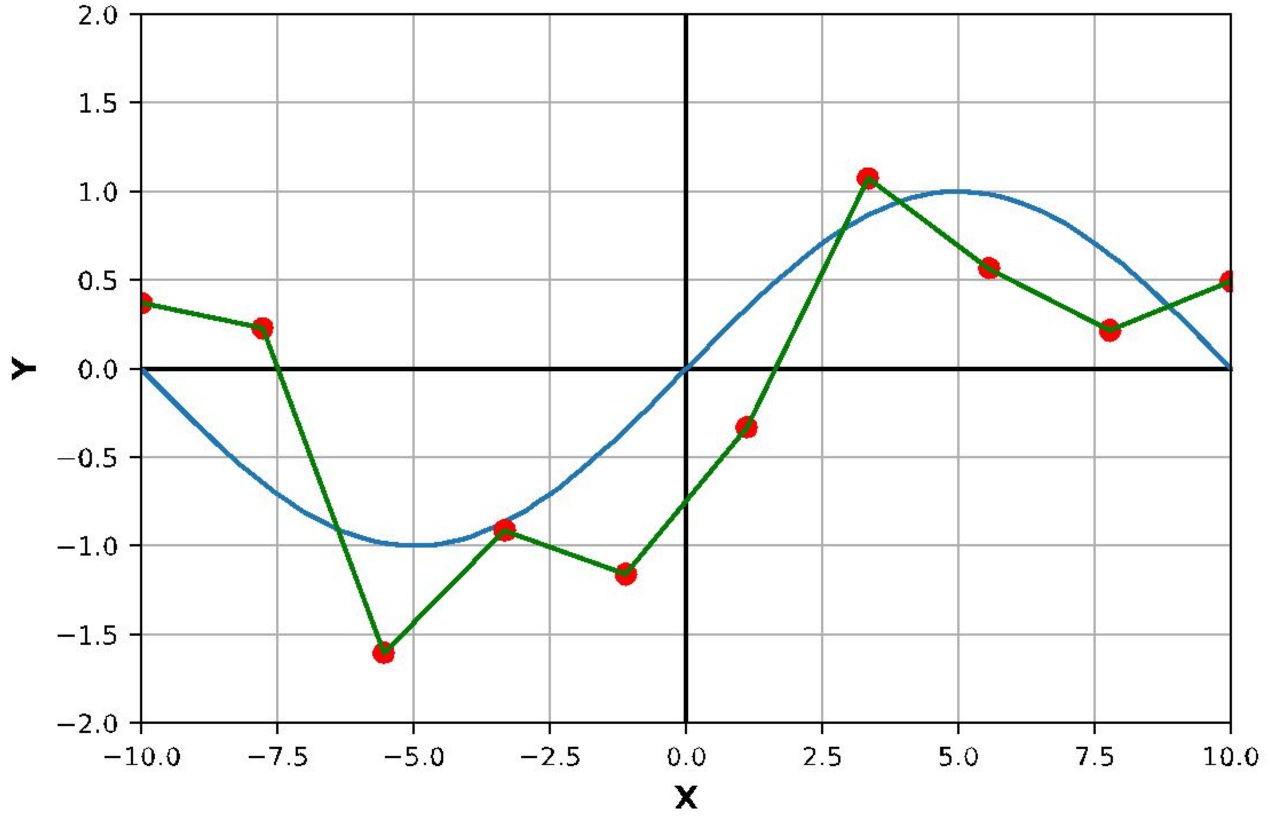


What is Bias and Variance?

$$\{1, x, x^2, x^3, x^4\}$$



Real Bad Overfit?



Bias-Variance Tradeoff

- So far we have minimized the error (loss) with respect to **training data**
  - Low training error does not imply good expected performance: **over-fitting**
- We would like to reason about the **expected loss (Prediction Risk)** over:
  - Training Data: $\{(y_1, x_1), \ldots, (y_n, x_n)\}$
  - Test point: (y_*, x_*)

Bias-Variance Tradeoff

To formally state the bias-variance tradeoff for regression problems, we consider the following setup (which is an extension of the beginning paragraph of Section 8.1).

- Draw a training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$ such that $y^{(i)} = h^\star(x^{(i)}) + \xi^{(i)}$ where $\xi^{(i)} \in N(0, \sigma^2)$.
- Train a model on the dataset S, denoted by $\hat{h}_S$.
- Take a test example (x, y) such that $y = h^\star(x) + \xi$ where $\xi \sim N(0, \sigma^2)$, and measure the expected test error (averaged over the random draw of the training set S and the randomness of ξ)[56]

$$\mathrm{MSE}(x) = \mathbb{E}_{S,\xi}[(y - h_S(x))^2] \tag{8.2}$$

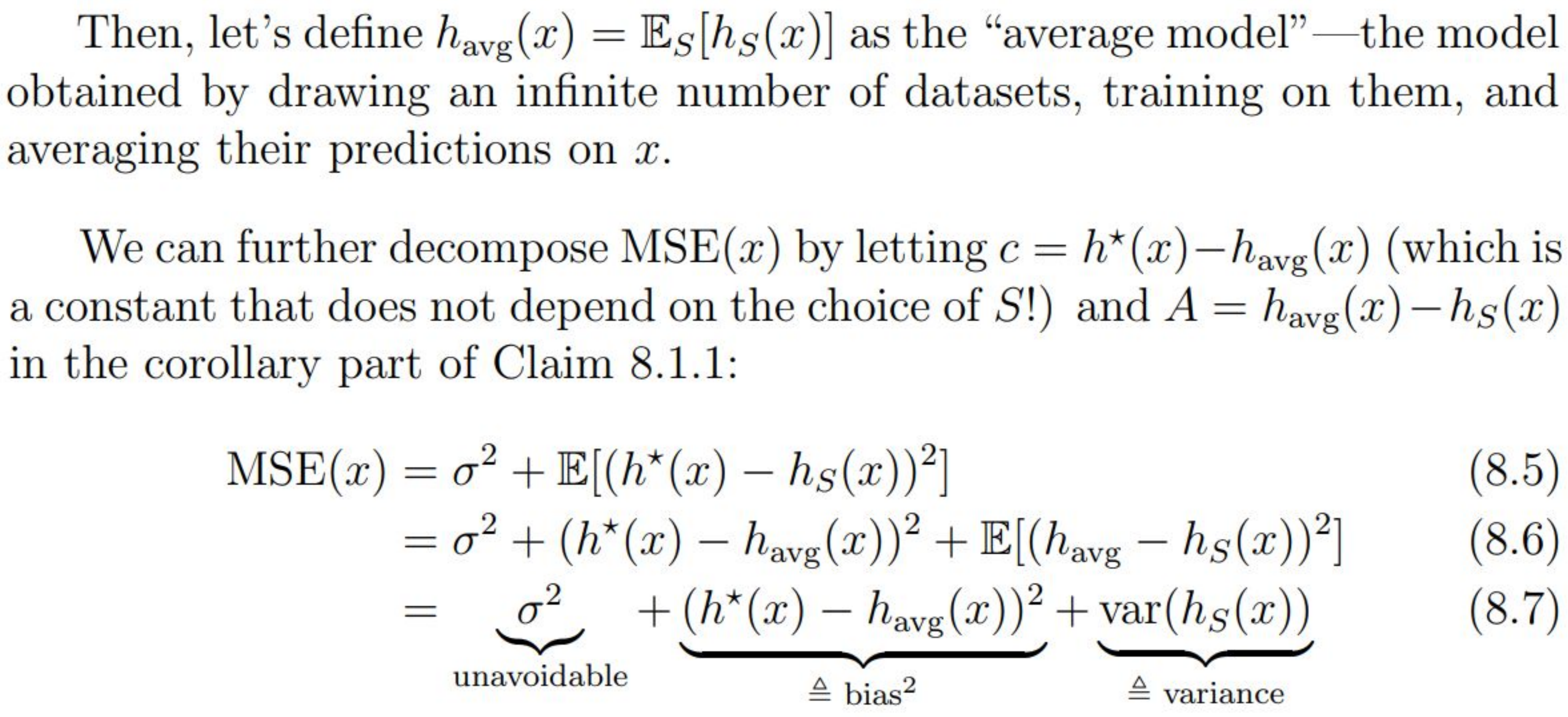
Bias-Variance Tradeoff

Claim 8.1.1: Suppose A and B are two independent real random variables and $\mathbb{E}[A] = 0$. Then, $\mathbb{E}[(A+B)^2] = \mathbb{E}[A^2] + \mathbb{E}[B^2]$.

Using Claim 8.1.1 with $A = \xi$ and $B = h^\star(x) - \hat{h}_S(x)$, we have

$$\begin{aligned}
\mathrm{MSE}(x) = \mathbb{E}[(y - h_S(x))^2] &= \mathbb{E}[(\xi + (h^\star(x) - h_S(x)))^2] && (8.3)\\
&= \mathbb{E}[\xi^2] + \mathbb{E}[(h^\star(x) - h_S(x))^2] && \text{(by Claim 8.1.1)}\\
&= \sigma^2 + \mathbb{E}[(h^\star(x) - h_S(x))^2] && (8.4)
\end{aligned}$$

Bias-Variance Tradeoff



Then, let's define $h_{\text{avg}}(x) = \mathbb{E}_S[h_S(x)]$ as the "average model"—the model obtained by drawing an infinite number of datasets, training on them, and averaging their predictions on x.

We can further decompose $\text{MSE}(x)$ by letting $c = h^\star(x) - h_{\text{avg}}(x)$ (which is a constant that does not depend on the choice of S!) and $A = h_{\text{avg}}(x) - h_S(x)$ in the corollary part of Claim 8.1.1:

$$\begin{aligned} \text{MSE}(x) &= \sigma^2 + \mathbb{E}[(h^\star(x) - h_S(x))^2] && (8.5) \\ &= \sigma^2 + (h^\star(x) - h_{\text{avg}}(x))^2 + \mathbb{E}[(h_{\text{avg}} - h_S(x))^2] && (8.6) \\ &= \underbrace{\sigma^2}_{\text{unavoidable}} + \underbrace{(h^\star(x) - h_{\text{avg}}(x))^2}_{\triangleq\ \text{bias}^2} + \underbrace{\text{var}(h_S(x))}_{\triangleq\ \text{variance}} && (8.7) \end{aligned}$$

Bias-Variance Tradeoff

$$\mathrm{MSE}(x) = \underbrace{\sigma^2}_{\text{unavoidable}} + \underbrace{(h^\star(x) - h_{\text{avg}}(x))^2}_{\triangleq\ \text{bias}^2} + \underbrace{\mathrm{var}(h_S(x))}_{\triangleq\ \text{variance}}$$

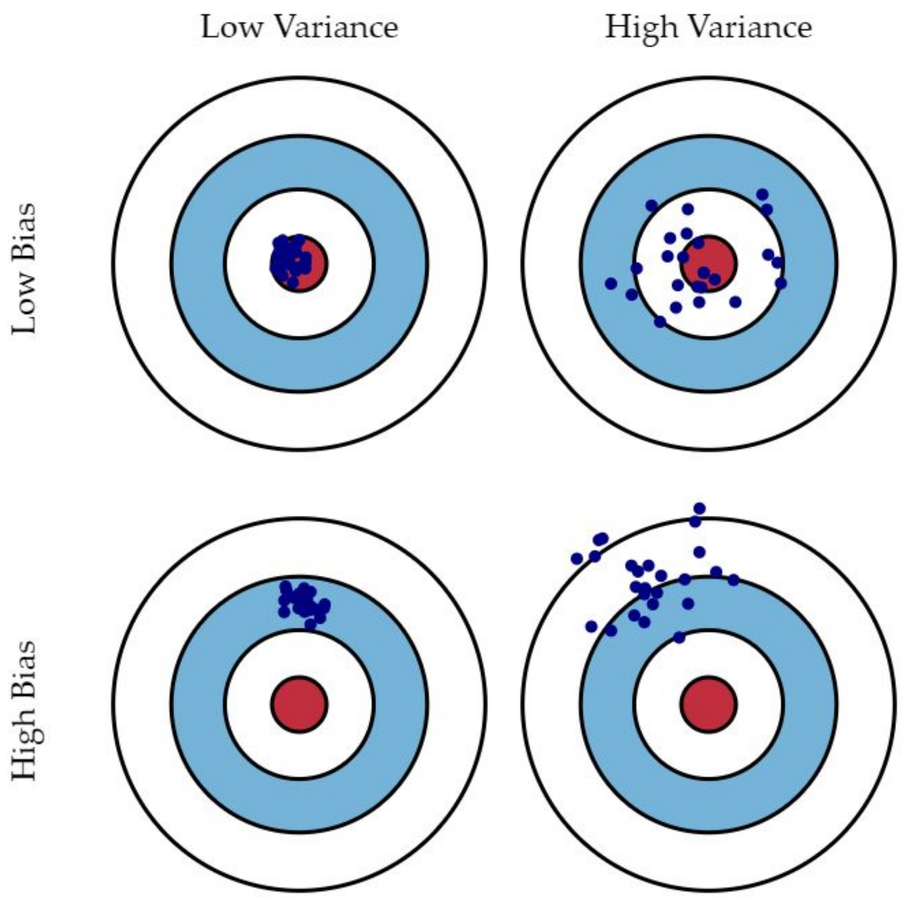
Recall that h_{avg} can be thought of as the best possible model learned even with infinite data. Thus, the bias is not due to the lack of data, but is rather caused by that the family of models fundamentally cannot approximate the $h^\star$.

The variance term captures how the random nature of the finite dataset introduces errors in the learned model. It measures the sensitivity of the learned model to the randomness in the dataset. It often decreases as the size of the dataset increases.

There is nothing we can do about the first term σ^2 as we can not predict the noise ξ by definition.

Bias and Variance

We can plot four different cases representing combinations of both high and low bias and variance.



from https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias Variance Plot

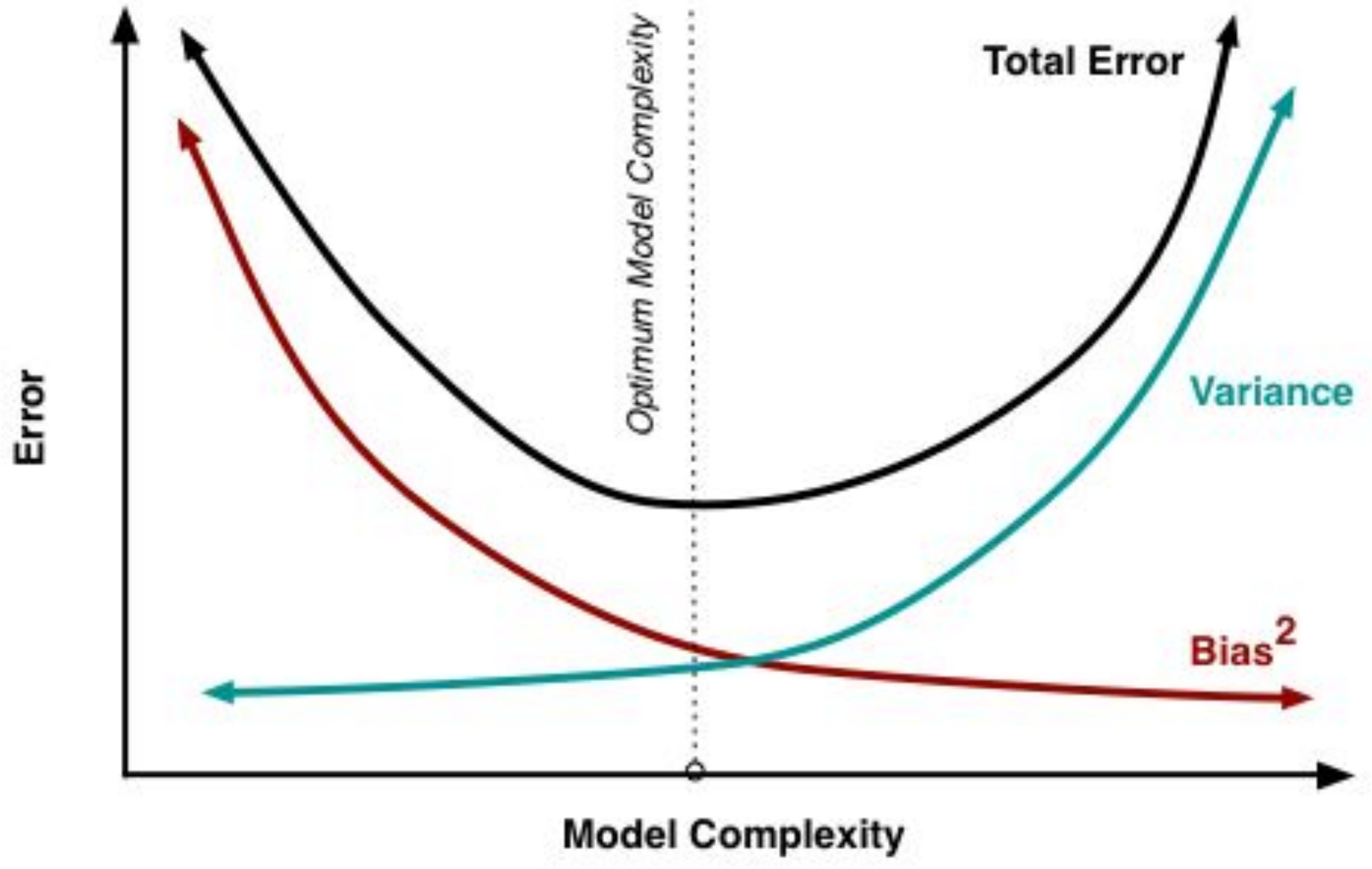


Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

Managing Bias Variance

The rule is we should seek **lower bias** and **lower variance**.

In general what we really care about is **lower overall error**, not the specific decomposition.

At its root, dealing with bias and variance is really about **dealing with over- and under-fitting**.

What to do in practice?

Keep a validation/test datasets to check the model performance.

Data : manipulate the data through data augmentation and resampling techniques.

Complexity: manipulate the model complexity by adding or reducing parameters.

Learned model: use regularization techniques.

Practice Time!

Regularization

Slides from: https://harvard-iacs.github.io/2019-CS109A/lectures/lecture8/presentation/Lecture8a_Regularization.pptx

Regularization: An Overview

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

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 ***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:

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

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

- 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, LASSO - Computational complexity

Solution to ridge regression:

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

The solution to the LASSO regression:

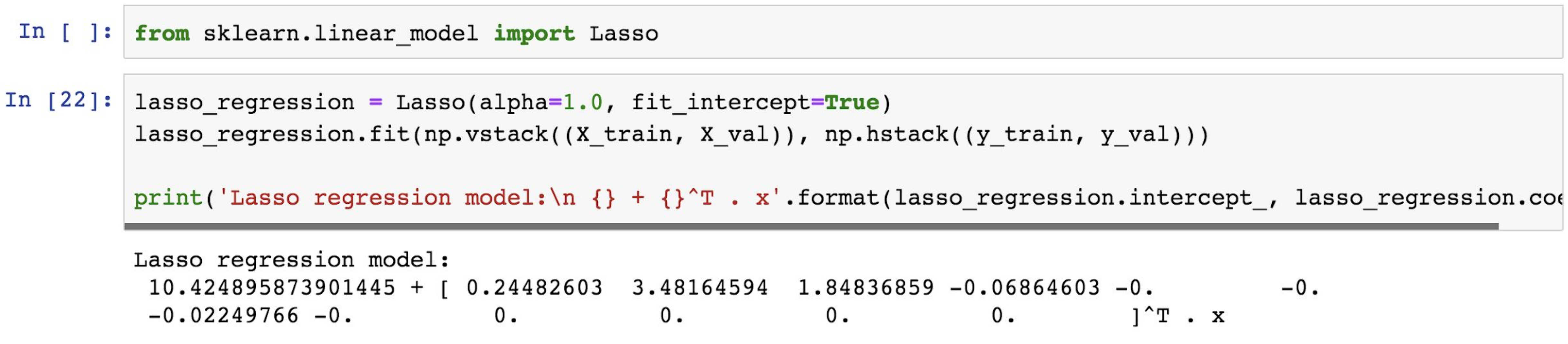
LASSO has no conventional analytical solution, as the L1 norm has no derivative at 0. We can, however, use the concept of subdifferential or subgradient to find a manageable expression.

Regularization Parameter with a Validation Set

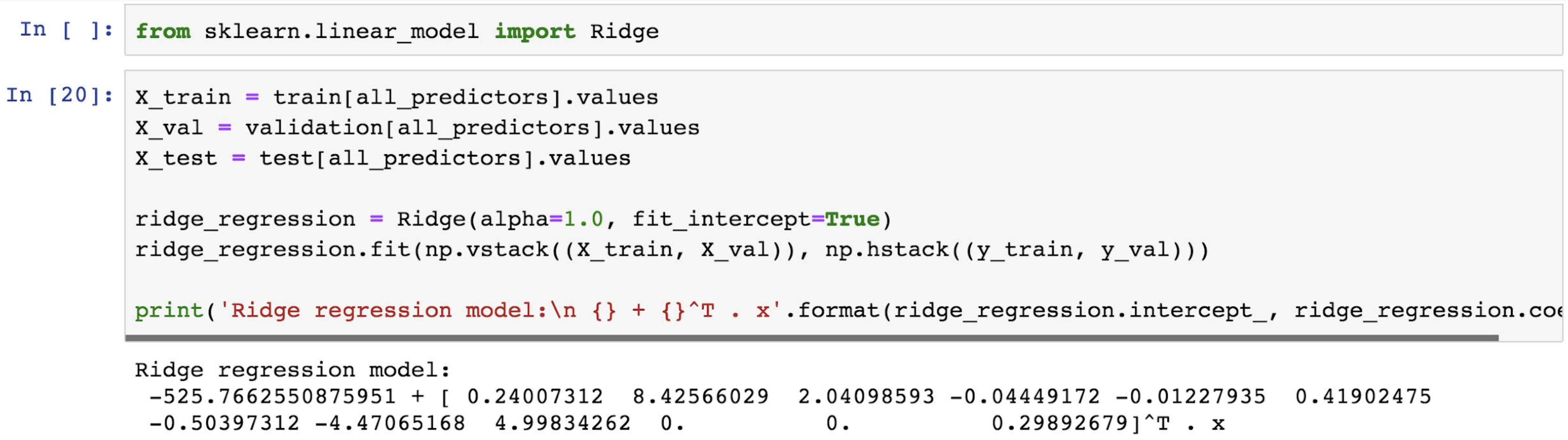
The solution of the Ridge/Lasso regression involves three steps:

- 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*

Examples

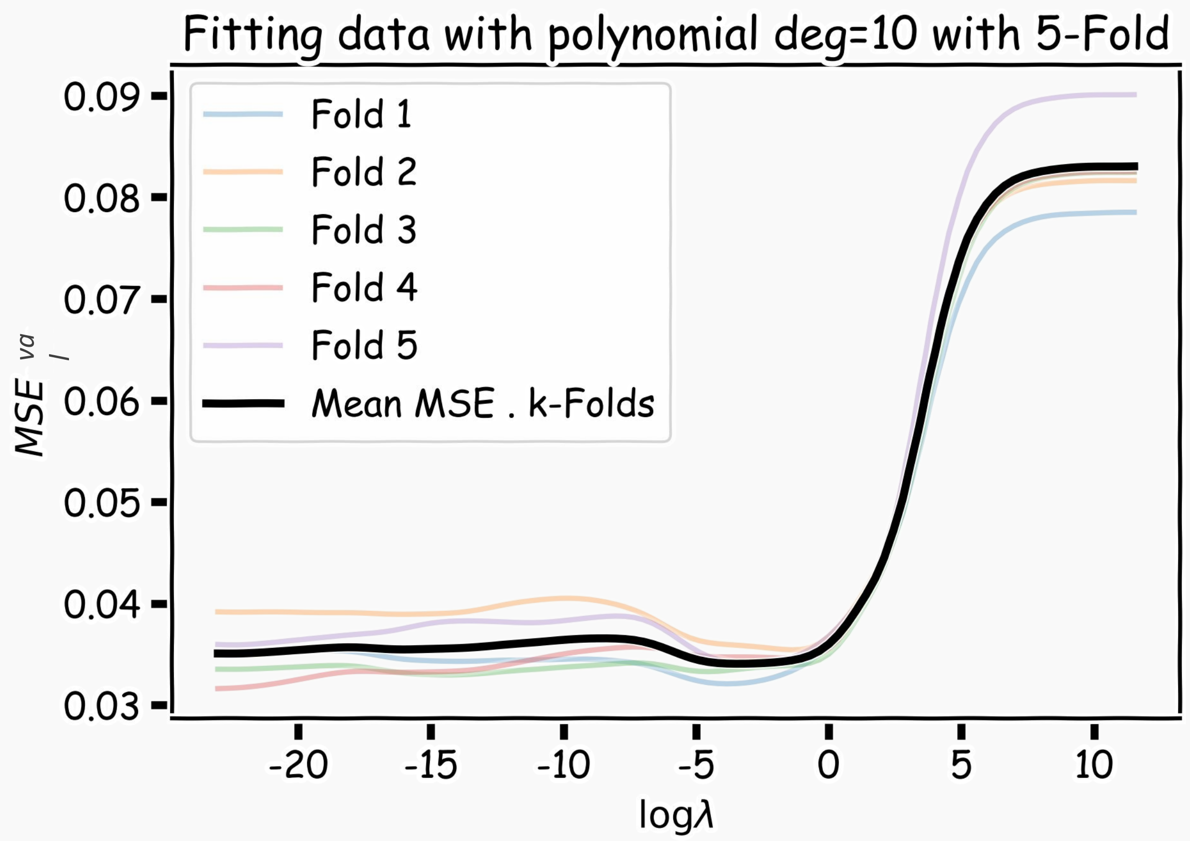


```
In [ ]:  from sklearn.linear_model import Lasso

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 [ ]:  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
```

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.

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