Linear Regression

Jing Sun

Next Lecture – Nonlinear Regression by David :)

Acknowledgement:

Dr. Merve Gürel (regressionS),

Dr. Tom Viering (perceptron),

Dr. David Tax (all the good old stuff and discussions)

Agenda

- Linear Regression:

  Ordinary Least Squares (OLS); R^2; Absolute Loss and Huber Loss.
- Overfitting and Bias-Variance Tradeoff.
- Regularizations:

  Ridge a.k.a. L2 and Lasso a.k.a. L1.
- A little bit of the Classifiers:

  (Multiclass) Logistic Regression and Perceptron (as there are some links).
- Bayesian Linear Regression (also prepare you for the next lecture).

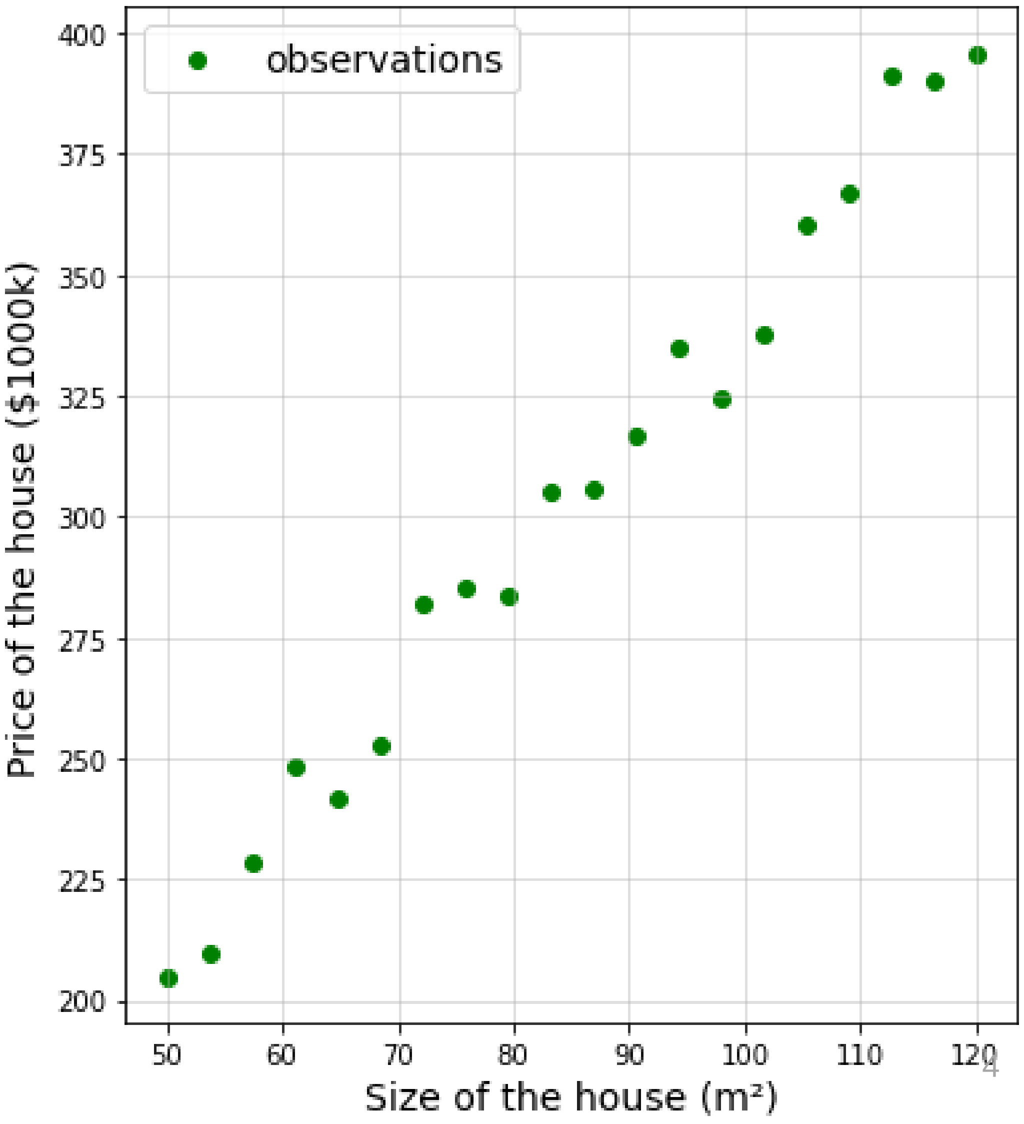
TU Delft

OLS Linear Regression

Linear Regression

- Given a set of features $x \in \mathbb{R}^d$,
  we predict a response variable $y \in \mathbb{R}$

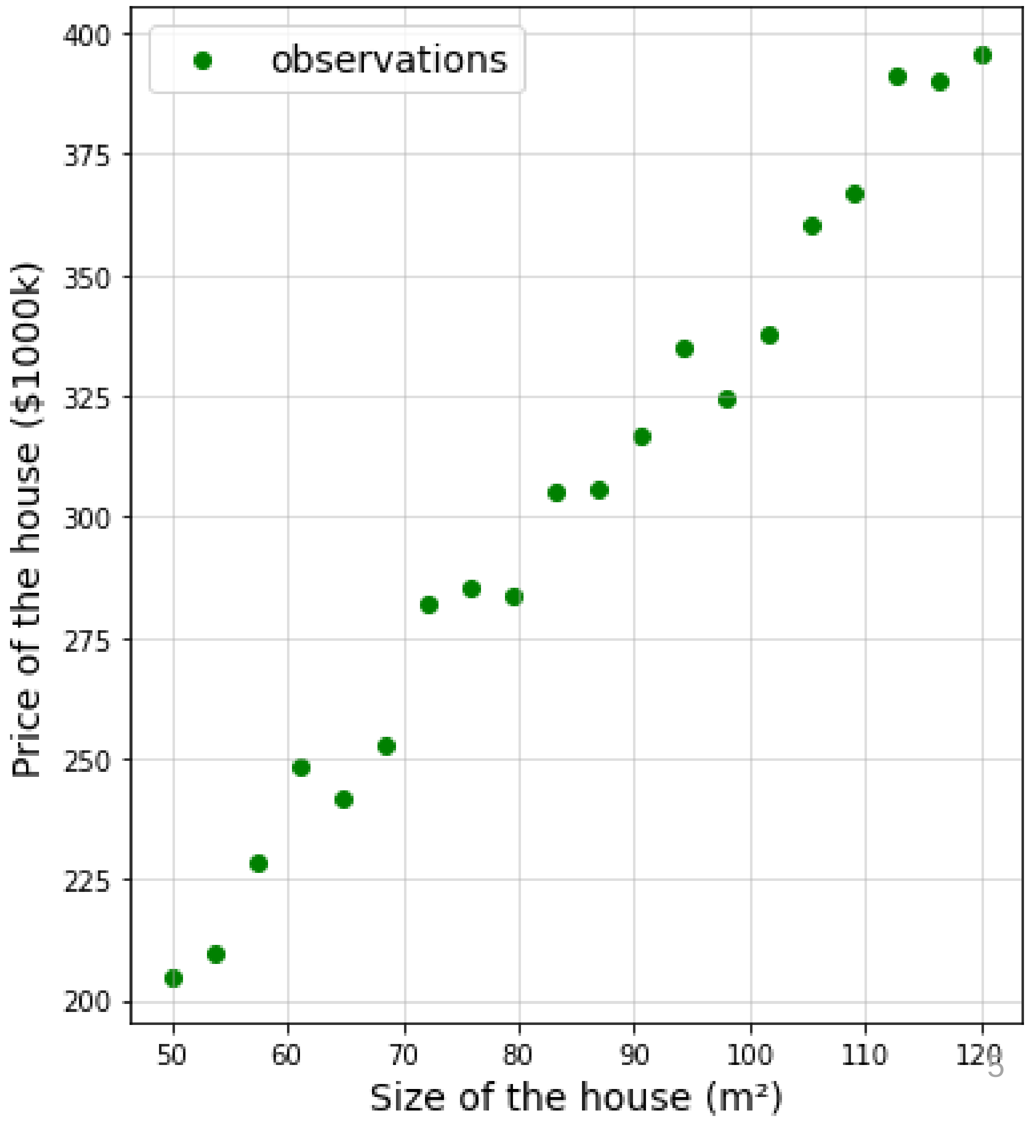
- Let's look at an example,
  input (x-axis): size of the house
  output (y-axis): price of the house



Linear Regression

- How to predict a new house's price given its size (an unseen x)?

- We can fit a linear function $f()$ to map x to y using the observations:



Linear Regression

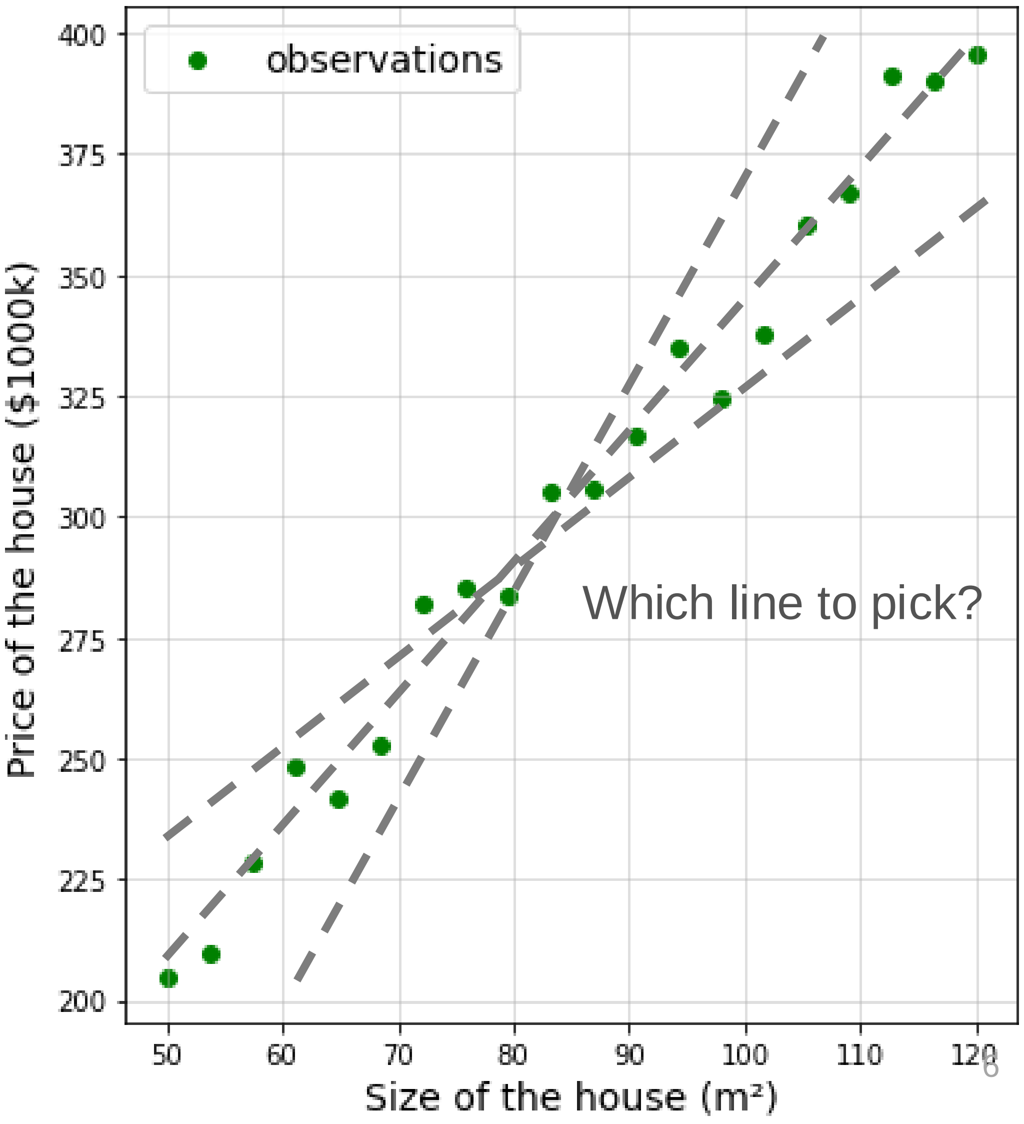
$$y = \beta_1 x_1 + \quad \beta_0, \qquad \mathrm{d} = 1$$

slope intercept

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix}, \qquad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$y = x^T \beta$$

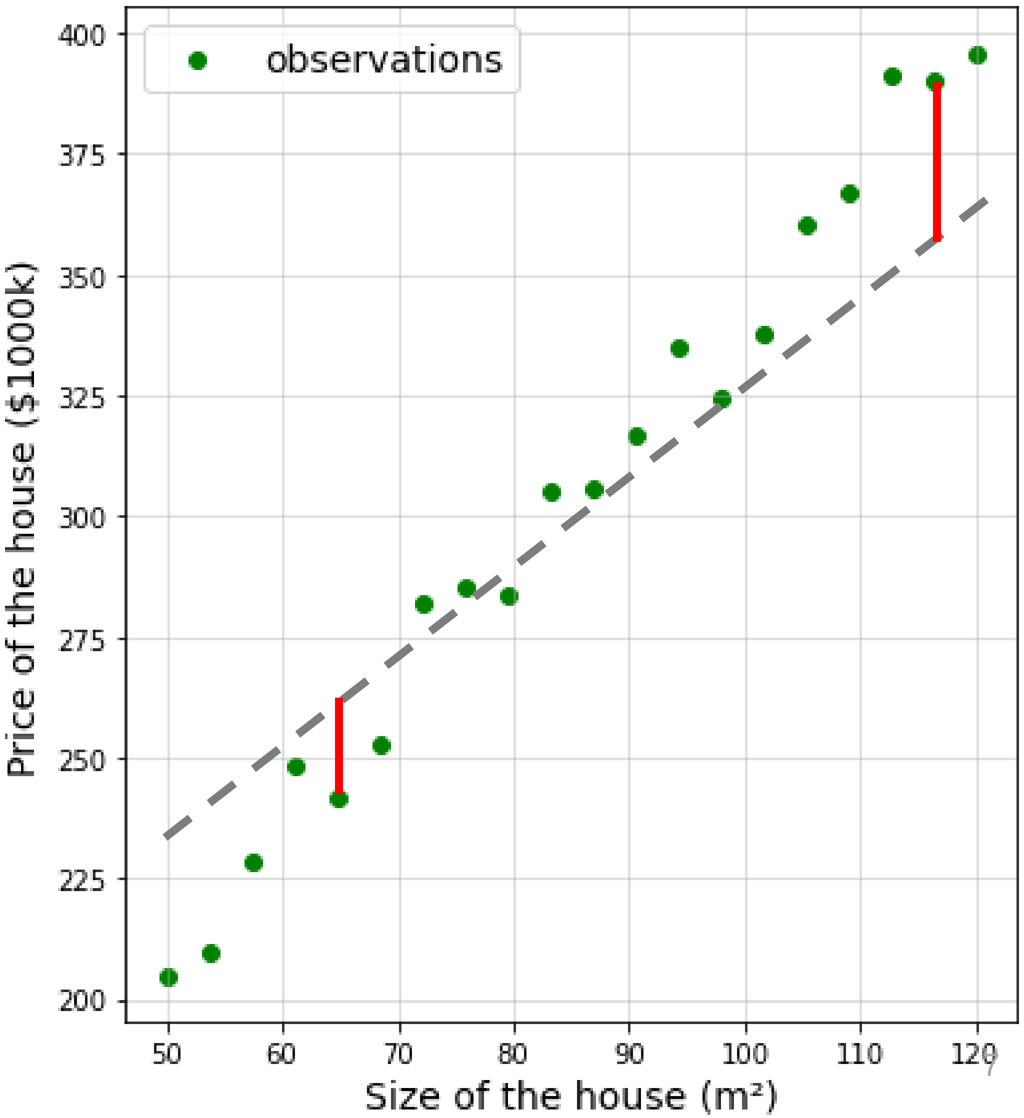
- How to estimate β?



Residuals

- Well, we want β to make y and $\hat{y}$ as close as possible, that is to say, to minimize the "residual".

- What is residual r_n?
- It is the **difference** between the **observed value** and the **predicted value** for a given data point.
- $r_n = y_n - \hat{y}_n = y_n - x_n^T \hat{\beta}$



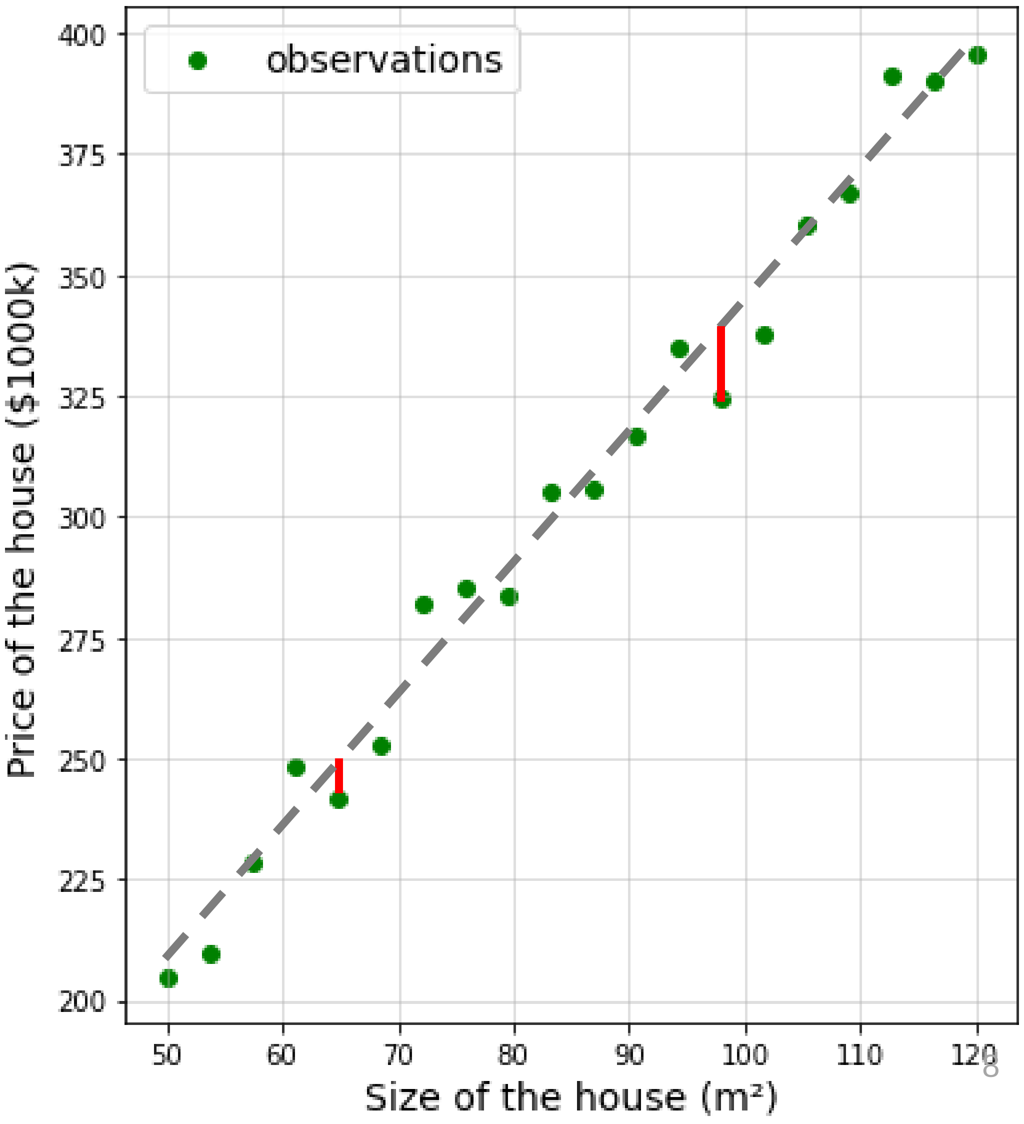
TU Delft

OLS

- Minimize the residual sum of squares!

- $RSS = \sum_{n=1}^{N}(r_n)^2 = \sum_{n=1}^{N}(y_n - \hat{y}_n)^2$

- sensitive to outliers



OLS

- $X = \begin{bmatrix} \boxed{\begin{matrix} 1 \\ 1 \\ 1 \\ 1 \end{matrix}} & \begin{matrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1d} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \cdots & x_{Nd} \end{matrix} \end{bmatrix}$ for the intercept, $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$,
- Parameter estimation:

$$\hat{\beta}_{OLS} = \underset{\beta \in \mathbb{R}^{d+1}}{argmin} \sum_{n=1}^{N} (y_n - x_n^T \beta)^2 = \underset{\beta \in \mathbb{R}^{d+1}}{argmin} (y - X\beta)^T (y - X\beta)$$

- Solution given at

$$\frac{\partial}{\partial \beta} (y - X\beta)^T (y - X\beta) = 0 \quad \Rightarrow \quad \hat{\beta}_{OLS} = (X^T X)^{-1} X^T y$$

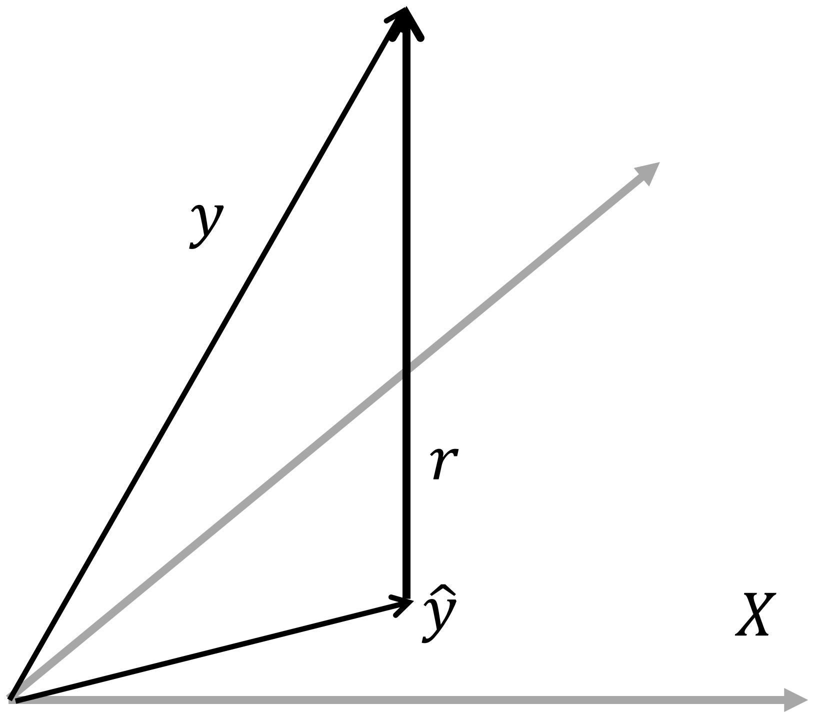
OLS – another angle

- Geometric Interpretation to OLS:

- Find the **orthogonal projection** of the y onto the d-dimensional subspace spanned by the X.

$$X^T\left(y - X\hat{\beta}_{\mathrm{OLS}}\right) = 0$$

$$\Rightarrow \quad \hat{\beta}_{OLS} = (X^T X)^{-1} X^T y$$



R^2: Coefficient of Determination

- The proportion of variability in Y that can be explained by the regression.

Total Sum of Squares $$TSS = \sum (y_n - \bar{y})^2$$

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

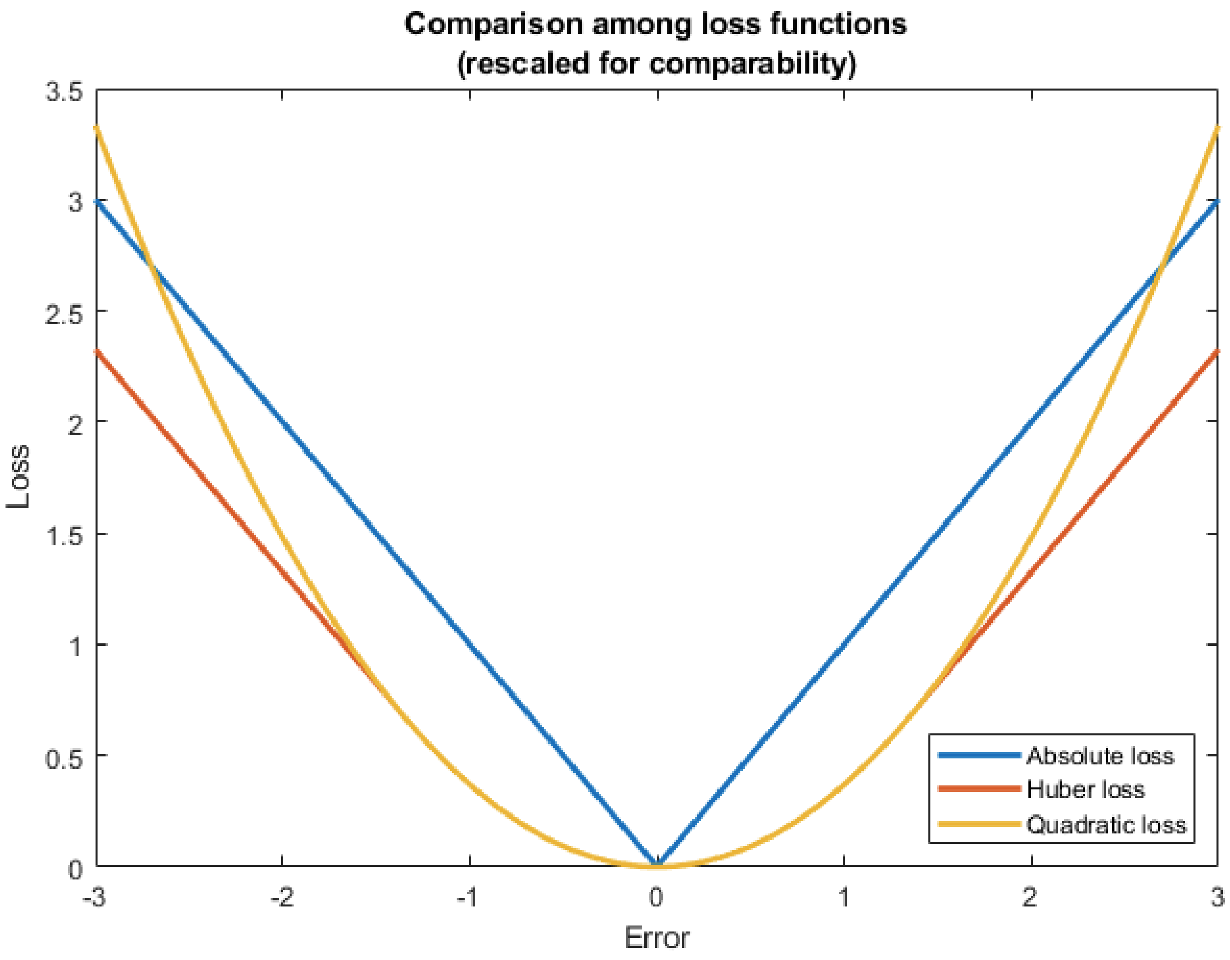
$$R^2 = \frac{variation(data) - variation(fit)}{variation(data)}$$

Other Loss Functions

- Absolute Loss= $|y - \hat{y}|$

- Huber Loss=
- $\begin{cases} \frac{1}{2}(y - \hat{y})^2, & if\,|y - \hat{y}| \leq \delta \\ \delta\left(|y - \hat{y}| - \frac{\delta}{2}\right), & otherwise \end{cases}$

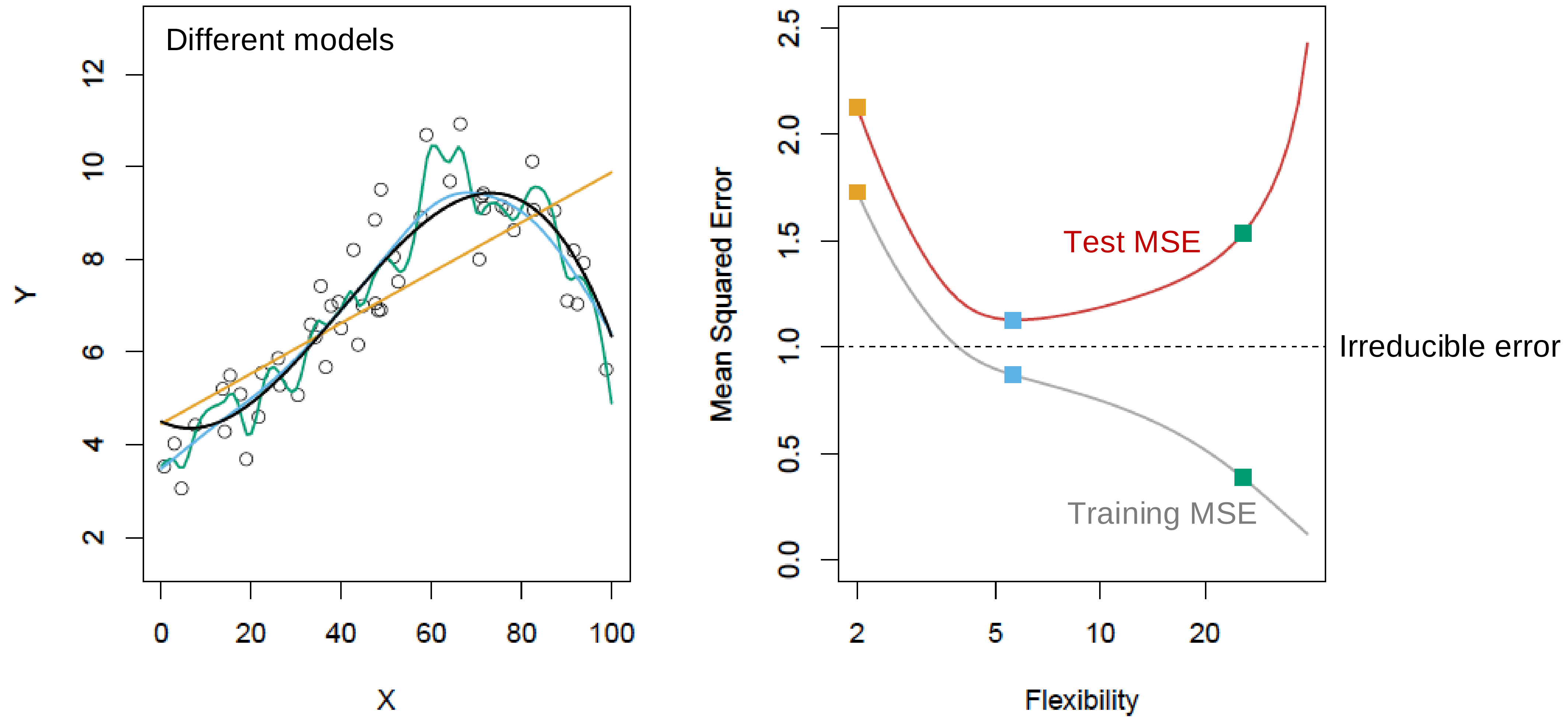
- Quadratic Loss: $(y - \hat{y})^2$



Taboga, 2021, https://www.statlect.com/glossary/loss-function

Overfitting and Bias-Variance Tradeoff

Bia-Variance Tradeoff



An introduction to statistical learning: with applications in Python Chapter 2.2.1 & 2.2.2

Bia-Variance Tradeoff

$$
\begin{aligned}
\mathrm{E}\left(Y - \hat{Y}\right) &= E\left[f(X) + \epsilon - \hat{f}(X)\right]^2 \\
&= \underbrace{\left[f(X) - \hat{f}(X)\right]^2}_{} \qquad\qquad + Var(\epsilon) \\
&= reducible\ error \qquad\qquad + irreducible\ error \\
&= \mathrm{Bias}\left(\hat{f}(X)\right)^2 + \mathrm{Var}\left(\hat{f}(X)\right) + \mathrm{Var}(\epsilon)
\end{aligned}
$$

- Do you remember the Bayes error rate?

-- the lowest possible error rate given the features.

-- analogous to the irreducible error.

Bia-Variance Tradeoff

- **Variance** refers to the amount by which $\hat{f}$ would **change** if we estimated it using a **different training data set**.

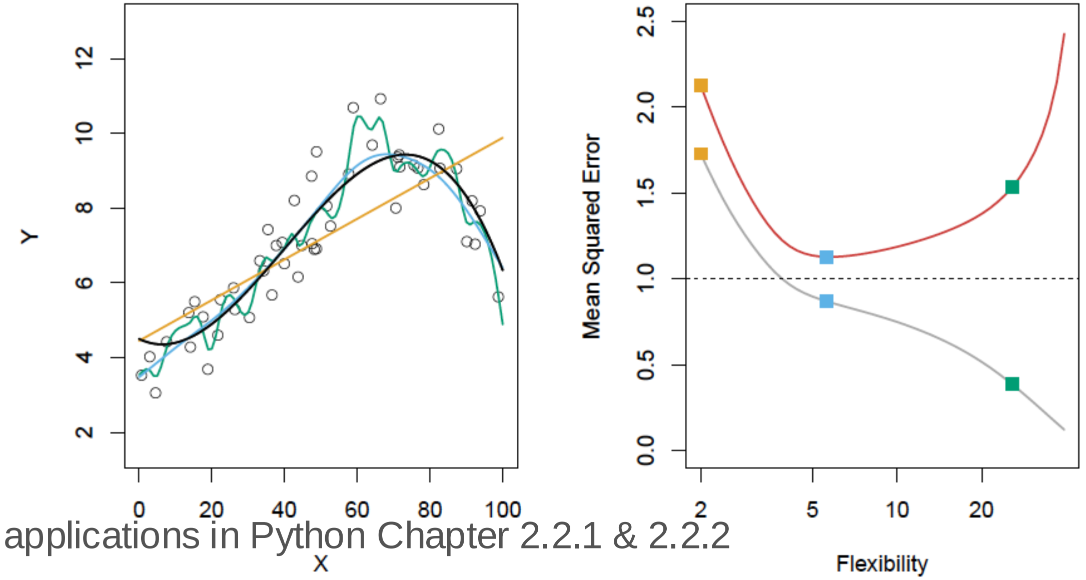
  High variance -- small changes in the training data can result in large changes in $\hat{f}$.

- **Bias** refers to the error that is introduced by **approximating a real-life problem**, which may be **extremely complicated**, by a **much simpler model**.

  The more we allow the model to accommodate to the training data set, the lower the bias.

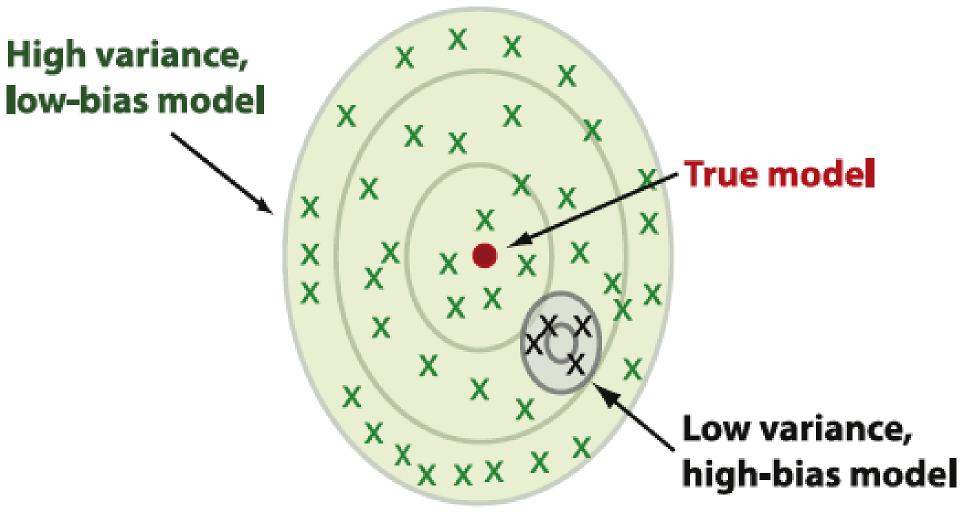
- Example: the flexible green curve

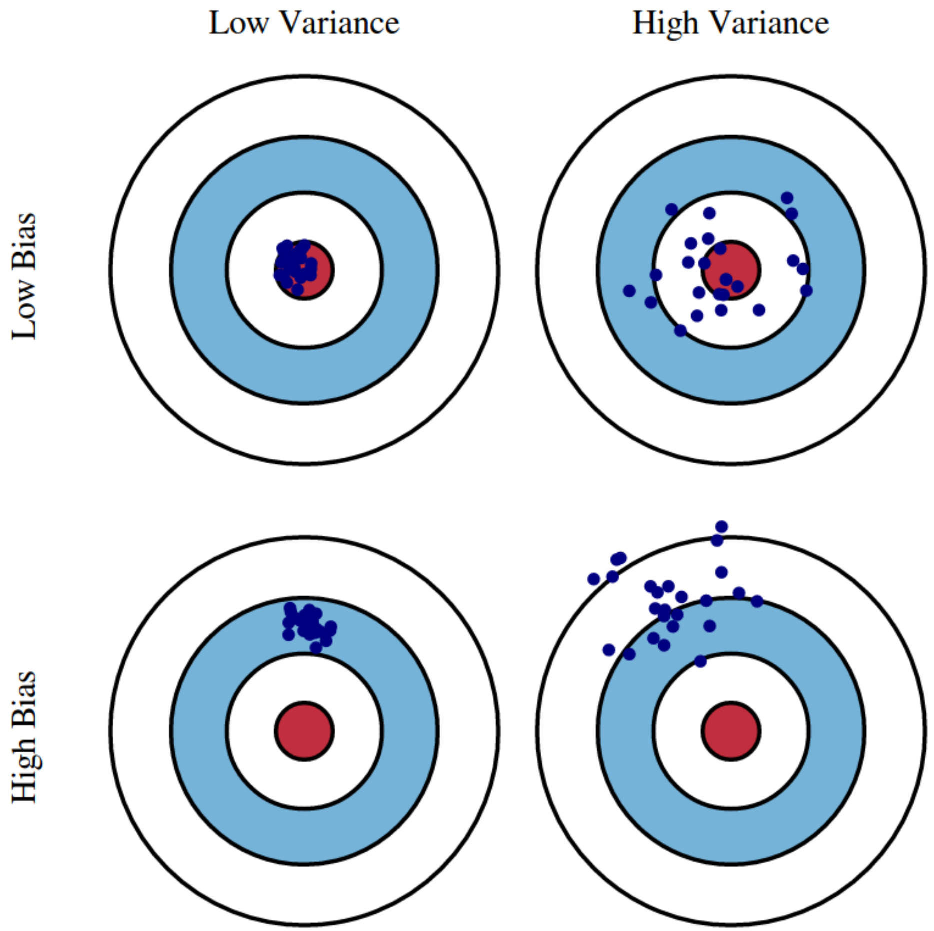
  (High Variance and Low Bias → Overfitting)



An introduction to statistical learning: with applications in Python Chapter 2.2.1 & 2.2.2

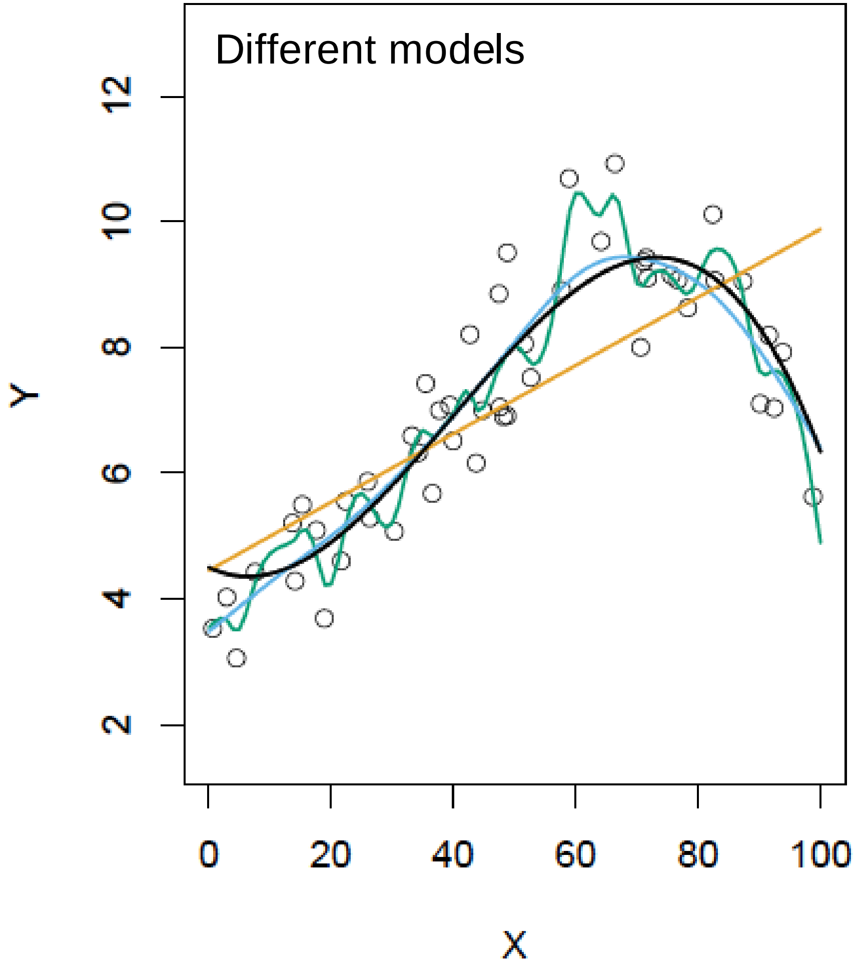
Bia-Variance Tradeoff

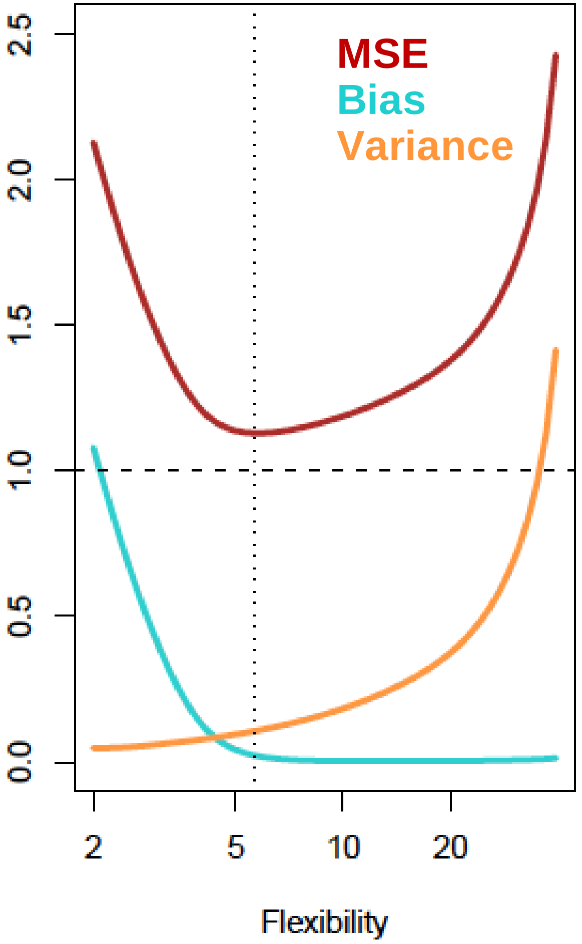




- Each point represents a model that has been fitted using some training data.

Bia-Variance Tradeoff



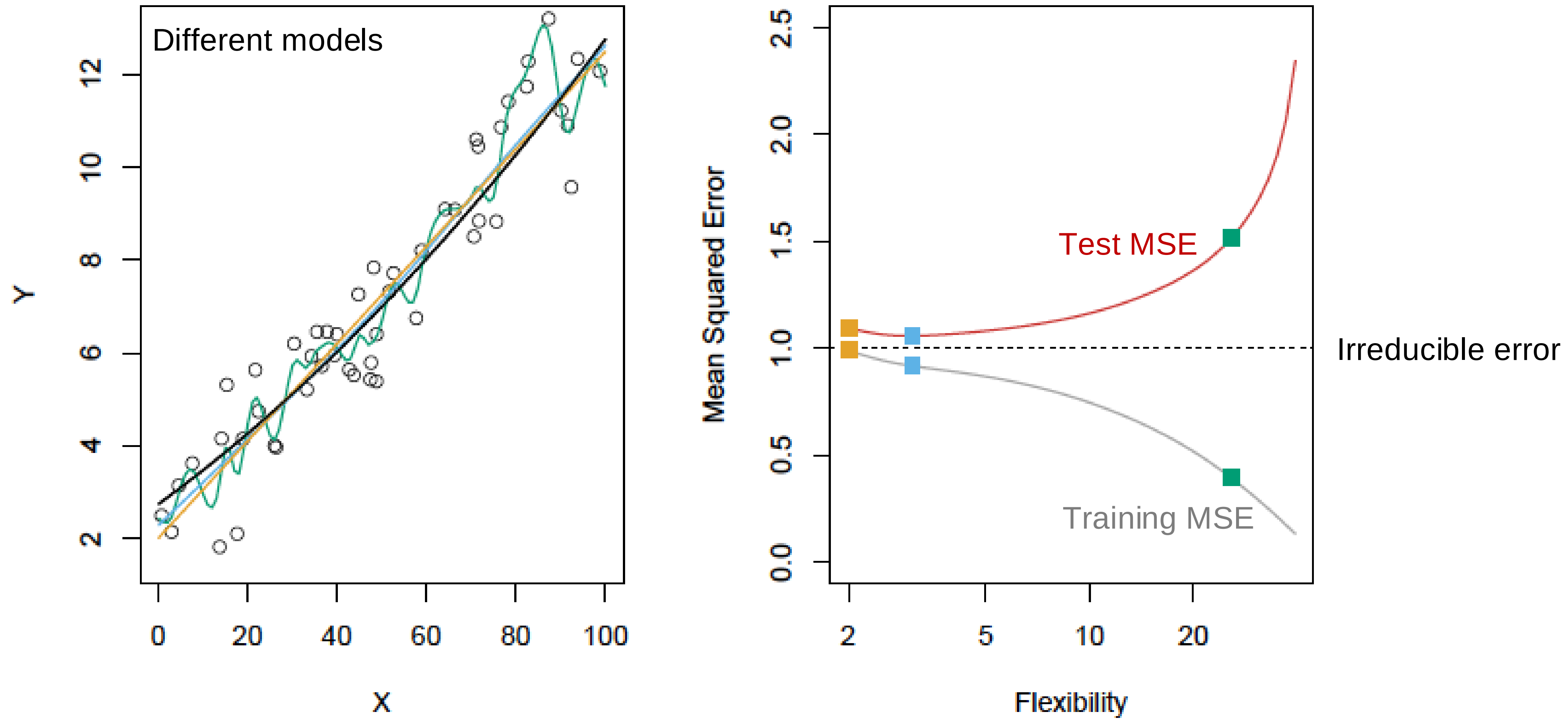


Irreducible error

TU Delft

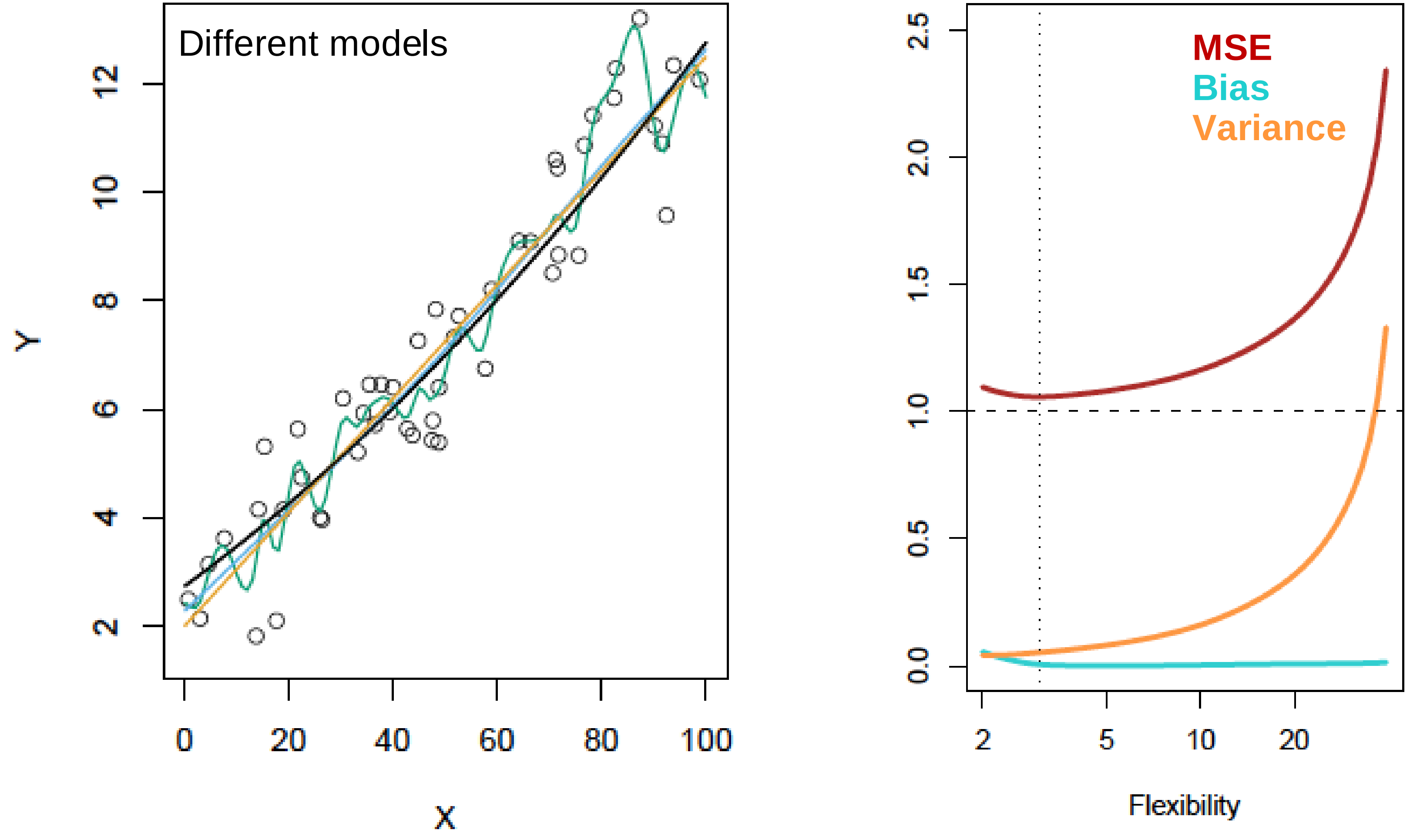
An introduction to statistical learning: with applications in Python Chapter 2.2.1 & 2.2.2

Bia-Variance Tradeoff



An introduction to statistical learning: with applications in Python Chapter 2.2.1 & 2.2.2

Bia-Variance Tradeoff



Irreducible error

An introduction to statistical learning: with applications in Python Chapter 2.2.1 & 2.2.2

Regularization

Ridge

- Avoid overfitting to the observed data, especially when working with a small data set.
- Worsen the predictions by punishing the model:

$$\hat{\beta}_{\text{Ridge}} = \underset{\beta}{\operatorname{argmin}} \sum_{n=1}^{N} (y_n - x_n^T \beta)^2 + \lambda \beta^T \beta \quad \text{L2}$$
$$= \underset{\beta}{\operatorname{argmin}} (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

- Ridge regression has a closed-form solution:

$$\hat{\beta}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T y$$

TU Delft

Lasso

- *Lasso: least absolute shrinkage and selection operator*

$$\hat{\beta}_{\text{Lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{n=1}^{N} (y_n - x_n^T \beta)^2 + \lambda \|\beta\|_1$$

L1

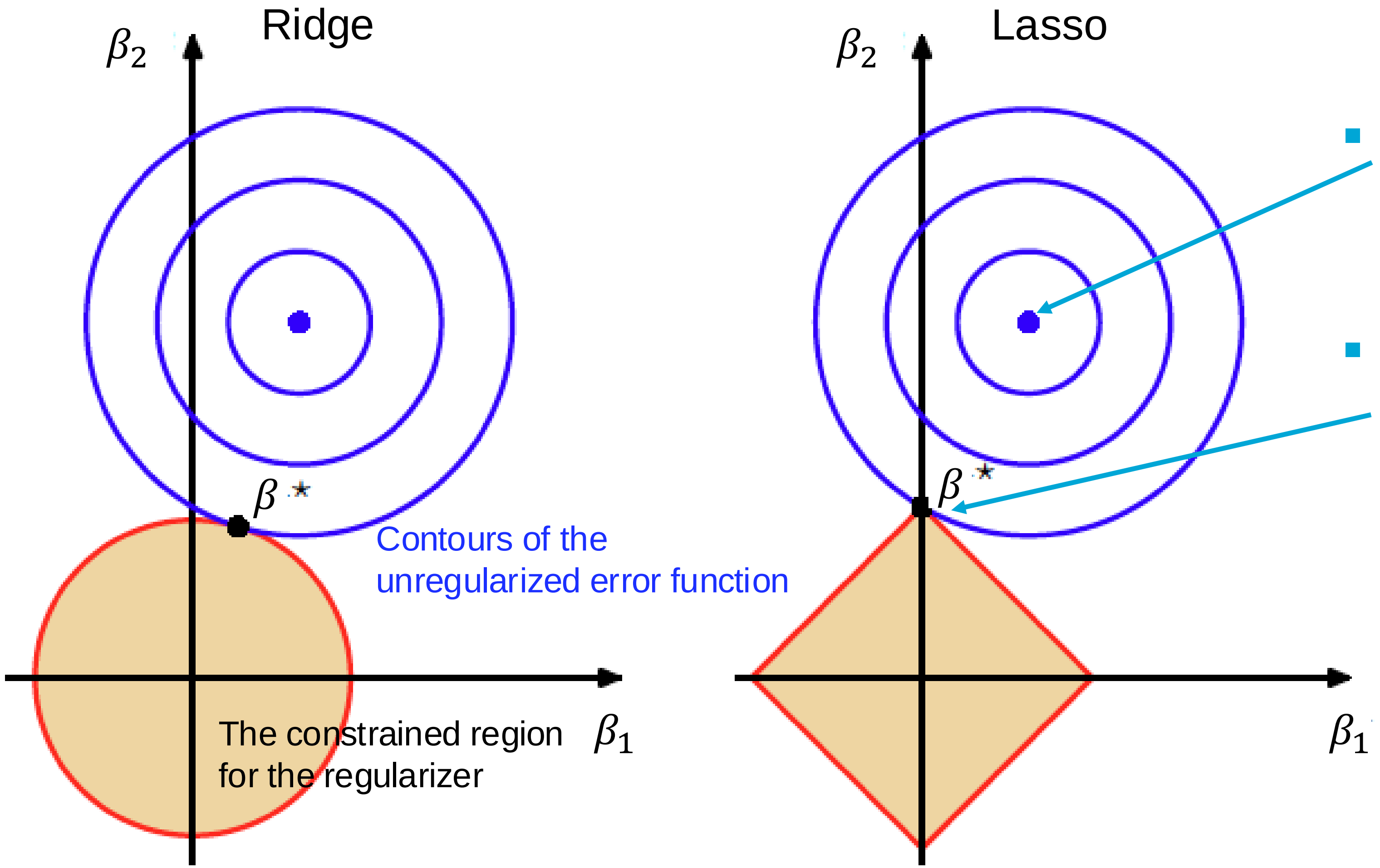
$$= \underset{\beta}{\operatorname{argmin}} (y - X\beta)^T (y - X\beta) + \lambda \|\beta\|_1$$

- No closed-form solution.

(e.g., coordinate descent. “The elements of statistical learning” Chapter 3.4.4 & 3.8.6 if interested)

- Introduce sparsity to the parameter space.
- Give penalty for having many predictors with small effects.

Regularizations



- Data term only (usual OLS estimate): all β_i non-zero
- Lasso gives a sparse solution, some β_i may be zero, in this example $\beta_1 = 0$

Classifiers: Logistic Regression and Perceptron

Logistic Regression

- Logistic regression model arises from the desire to model the **posterior probabilities of the K classes** via linear functions in x, while at the same time ensuring that they sum to one and remain in $[0, 1]$.

- -- Two classes

- -- More than two classes

The Bayes Rule if you still remember…

likelihood *prior*

posterior

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

evidence

Logistic Regression

- Consider the case of two classes,

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}$$

- We define $a = \ln\left(\frac{p(C_1|x)}{p(C_2|x)}\right) = ln\frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$

The inverse of the logistic sigmoid, i.e., the logit function, a.k.a. the log odds.

$$p(C_1|x) = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

The logistic ***sigmoid*** *(S-shaped) function.*

Logistic Regression

- Consider the case $K > 2$ classes,

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)}$$

$$a_k = \ln p(x|C_k)p(C_k)$$

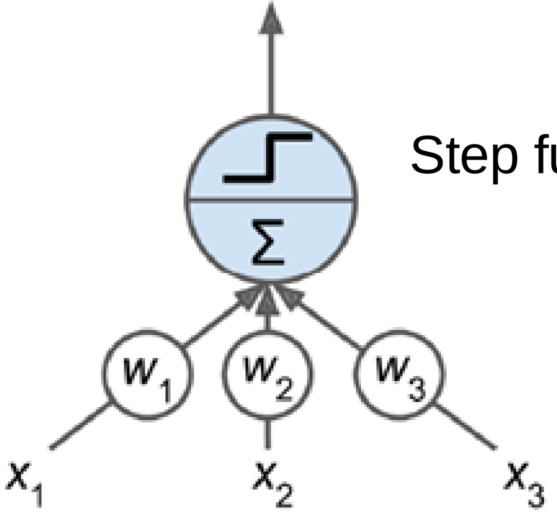
$$p(C_k|x) = \frac{exp(a_k)}{\sum_j exp(a_j)} = \sigma(a)$$

- The *normalized exponential,* a.k.a. *the softmax function,* is a multiclass generalization of *the logistic sigmoid*.

Perceptron

- Invented in 1957 by Frank Rosenblatt
- Linear classifier combined with a Heaviside/unit step function
- Step functions a.k.a. non-linearities

Output: $step(x^T W)$



Step function: $\mathrm{step}(z)$

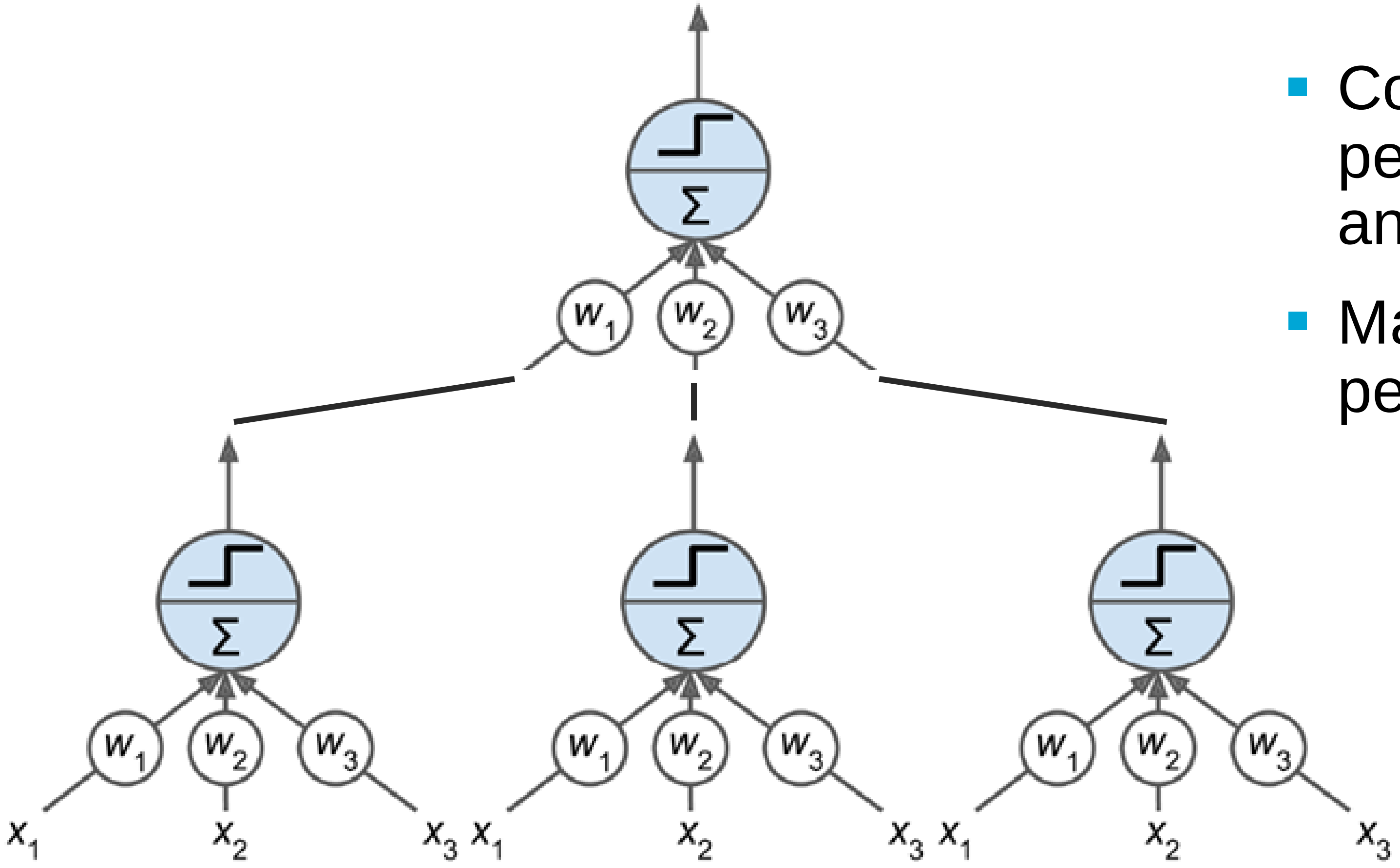
$$z = x^T W$$

$$Heaviside(z) = \begin{cases} 0 & if\ z < 0 \\ 1 & if\ z \geq 0 \end{cases}$$

- For regression?
- What if we use the sigmoid function as the step function?

TUDelft

Multilayer Perceptron (MLP)



- Connect the output of one perceptron to the inputs of another perceptron
- Make multiple layers, stacking perceptrons on top of each other.

> ➢ For multiclass classification?
>
> Softmax?

Bayesian Linear Regression

Look at OLS again from a probabilistic perspective

- We assume:

$$y_n = {x_n}^T\beta + \epsilon, \qquad where\ \epsilon\ i.i.d\ with\ \mathcal{N}(0, \sigma_\epsilon^2)$$

- Making assumption that data points are drawn independently from the above distribution, the likelihood function is:

$$p(y|X, \beta, \sigma_\epsilon^2) = \prod_{n=1}^{N} p(y_n|x_n^T\beta, \sigma_\epsilon^2)$$

$$= \frac{1}{\sqrt{(2\pi)^N \sigma_\epsilon^{2N}}} \exp(-\frac{1}{2\sigma_\epsilon^2}(y - X\beta)^T(y - X\beta))$$

TU Delft

Pattern Recognition and Machine Learning (Bishop 2006) Chapter 3.1.1 for full derivation

Look at OLS again from a probabilistic perspective

- Estimate parameters where $p(y|X,\beta,\sigma_\epsilon^2)$ is maximized:

$$\hat{\beta}_{ML} = \underset{\beta}{\operatorname{argmax}}\ p(y|X,\beta,\sigma_\epsilon^2)$$

- We get:

$$\hat{\beta}_{ML} = (X^T X)^{-1} X^T y$$

- Similarly, we can maximize the log likelihood function with respect to the σ_ϵ^2.

$$\hat{\sigma}_{ML}^2 = \frac{1}{N} \sum_{n=1}^{N} (r_n)^2$$

Look at OLS again from a probabilistic perspective

- Recall

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T y$$

- We see

$$\hat{\beta}_{ML} = \hat{\beta}_{OLS}$$

- OLS could be motivated as the **maximum likelihood** solution under an assumed Gaussian noise.

- What is the limitation? -- There is no representation of our **uncertainty**.

Bayesian Linear Regression

- The aim is not to find the best “single” value for the parameters, but rather to determine their posterior distribution.

- Recall the Bayes rule again,

$$\underbrace{p(A|B)}_{\text{posterior}} = \frac{\overbrace{p(B|A)}^{\text{likelihood}}\overbrace{p(A)}^{\text{prior}}}{\underbrace{p(B)}_{\text{evidence}}}$$

$$p(\beta|D) = \frac{p(D|\beta)p(\beta)}{p(D)} \qquad D = \big((x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\big), x_n \in \mathbb{R}^d, y_n \in \mathbb{R}$$

TU Delft

Maximum a Posterior (MAP) Estimation

- We have $y \sim \mathcal{N}(X\beta, \sigma_\epsilon^2 I)$

$$\hat{\beta}_{MAP} = \underset{\beta}{\operatorname{argmax}}\, p(\beta | X, y)$$

Likelihood of the response given the predictors and the model

Prior probability of the model parameters (we can use prior such as Gaussian)

$$= \underset{\beta}{\operatorname{argmax}} \frac{p(y|X,\beta)\, p(\beta)}{p(y|X)}$$

a normalization constant

$$= \underset{\beta}{\operatorname{argmax}}\, p(y|X,\beta) p(\beta)$$

$$\hat{\beta}_{MAP} = \left(\frac{\sigma_\epsilon^2}{\sigma_\beta^2} I + X^T X\right)^{-1} X^T y$$

Try the derivation!

TU Delft