ECE 57000

Linear and logistic regression

Chaoyue Liu

Fall 2024

Linear model

Scenario (house price prediction problem):

Consider the problem of predicting house prices y, based on given data on area x_1, age x_2, # of bedrooms x_3, …, x_d.

Call the model as *f*:

$$f(x_1, \cdots x_d) = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + w_d * x_d + b$$

Prediction (output of *f*)	larger area x_1 means higher price	older x_2 means lower price	more beds means higher price	Bias term

Linear model

Scenario (house price prediction problem):

Consider the problem of predicting house prices y, based on given data on area x_1, age x_2, # of bedrooms x_3, …,x_d.

Call the model as *f*:

$$f(x_1, \cdots x_d) = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + w_d * x_d + b$$

- Coefficients $w_1, w_2, \cdots, w_d$ are called **weights**, b is called **bias**.
- They are also called **parameters**, which will be tuned (learned) during training with data.
- The model f is a parametric model

Linear model

$$f(x_1, \cdots x_d) = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + w_d * x_d + b$$

- Let $\mathbf{x}=(x_1, x_2, x_3, \cdots, x_d, 1)^T$, $\mathbf{w}=(w_1, w_2, w_3, \cdots, w_d, b)^T$

The model can be written in a more compact form:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

Note: we may also write $f_{\mathbf{w}}(\mathbf{x})$ as $f(\mathbf{w}; \mathbf{x})$

Linear model is linear in terms of its input **x**, as well as of its parameters **w**.

Now we have built the model, let's move on to train the model…

Linear regression

Train the model: to find an optimal set of parameters $\mathbf{w}^*$, s.t. the model $f_{\mathbf{w}^*}$ performs the best.

We must find a metric to measure the performance of $f_{\mathbf{w}}$ for different $\mathbf{w}$.

Intuition: for a given datapoint $(\mathbf{x}, y)$, the closer between $f_{\mathbf{w}}(\mathbf{x})$ and y, the better.

Loss: $\mathcal{L}(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - f_{\mathbf{w}}(\mathbf{x}_i)\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \mathbf{x}_i^T\mathbf{w}\right)^2$

$= \frac{1}{2n}\|\mathbf{y} - \mathrm{X}\mathbf{w}\|_2^2$

Mean Least Square (MSE)

X: design matrix, each row is $\mathbf{x}_i^T$;
$\mathbf{y}$: label vector, $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$

The best parameters $\mathbf{w}^*$ is obtained by minimizing the loss

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y} - \mathrm{X}\mathbf{w}\|_2^2$$

Luckily, we have a closed form solution for linear model

- How do you find maximum or minimum in calculus?

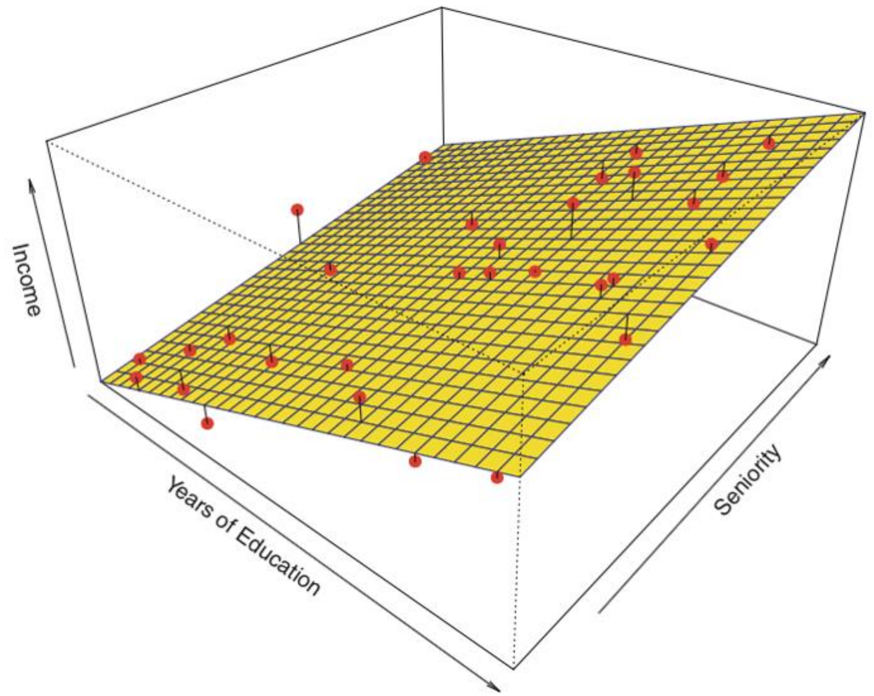
- Calculate gradient
  - $\nabla_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2 = \nabla_{\mathbf{w}} (\mathbf{y} - X\mathbf{w})^T (\mathbf{y} - X\mathbf{w})$
  - $= \left(2(\mathbf{y} - X\mathbf{w})^T(-X)\right)^T$
  - $= \left(2(-X^T)(\mathbf{y} - X\mathbf{w})\right)$
  - $= 2(-X^T\mathbf{y} + X^T X\mathbf{w})$

Derivation hints:
Use equivalence of $\|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^T\boldsymbol{v}$.
Then use matrix calculus (wikipedia reference).

- Set equal to zero and solve
  - $2(-X^T\mathbf{y} + X^T X\mathbf{w}) = 0$
  - $X^T X\mathbf{w} = X^T\mathbf{y}$
  - $\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}$

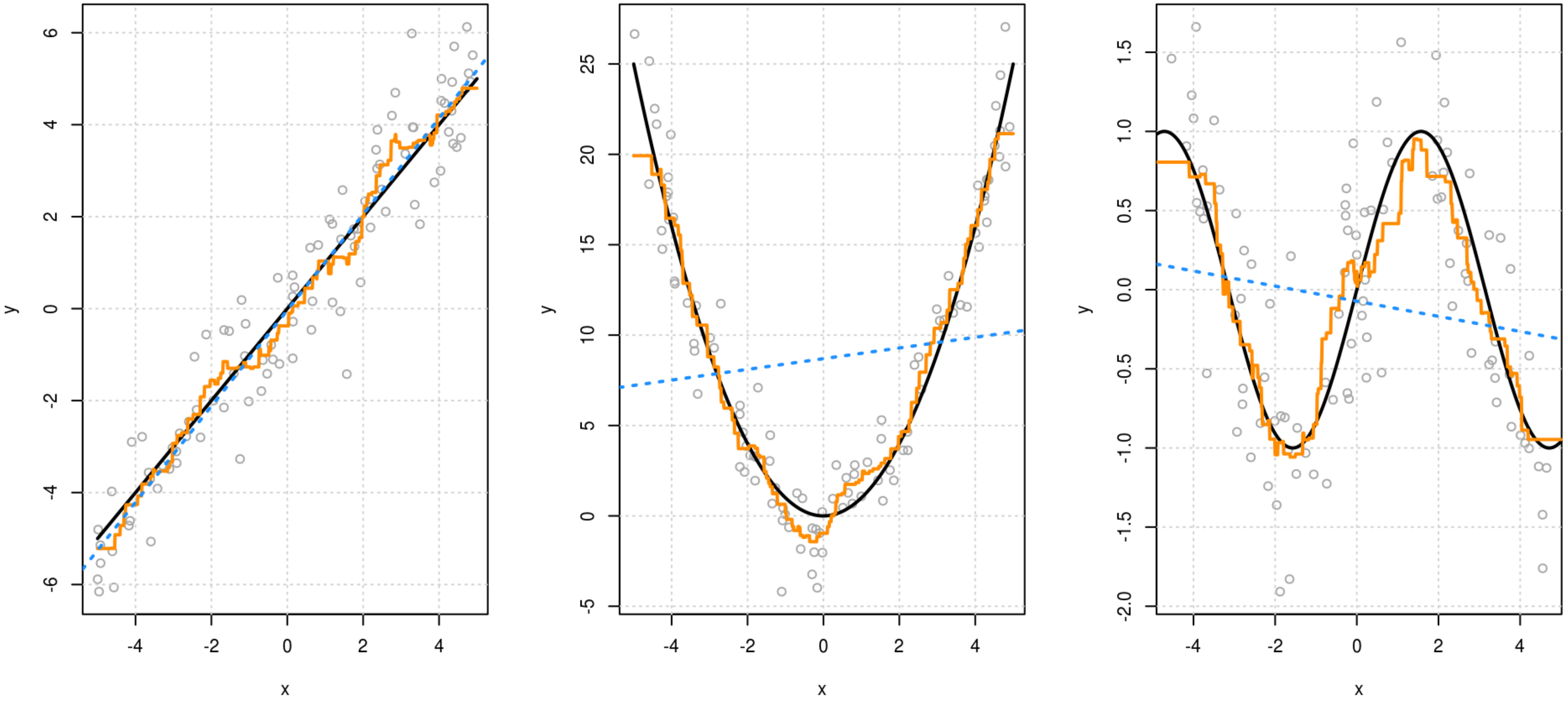
Time complexity: $\sim O(n^3)$

Linear regression models the output as a line (1D) or a hyperplane (>1D)



https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86

How does this compare to *k*-NN regression?
Linear regression is a much simpler function



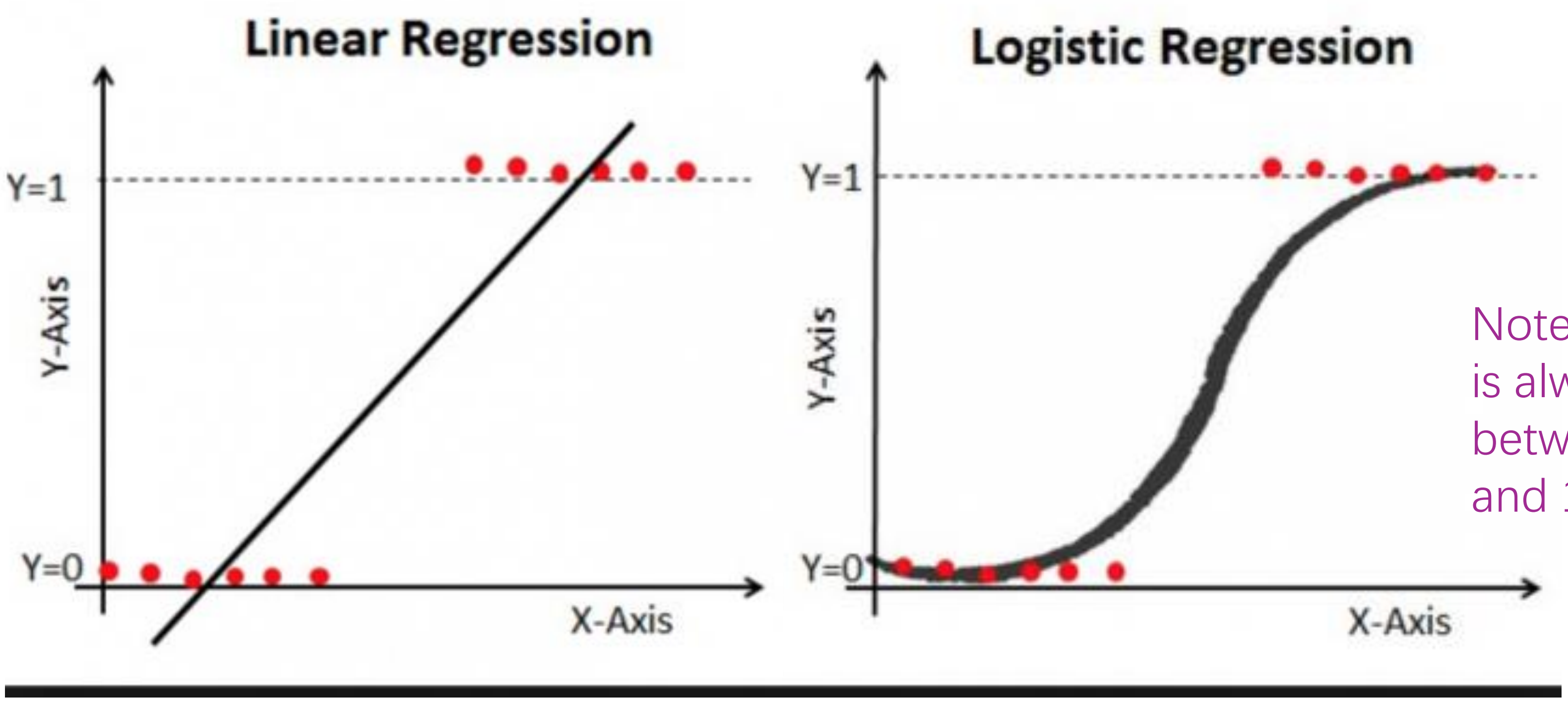
If true phenomena is linear (i.e., *assumption matches reality*), linear regression will do the best (left). However, if true phenomena is not linear, *k*-NN regression will perform better. (Black line is true function, dotted blue line is best linear approximation, and orange line is *k*-NN regression.)

https://daviddalpiaz.github.io/r4sl/knn-reg.html

Logistic regression

Logistic regression is used to solve binary classification problems

Setting: two classes $y \in \{0,1\}$.



Note that it is always between 0 and 1

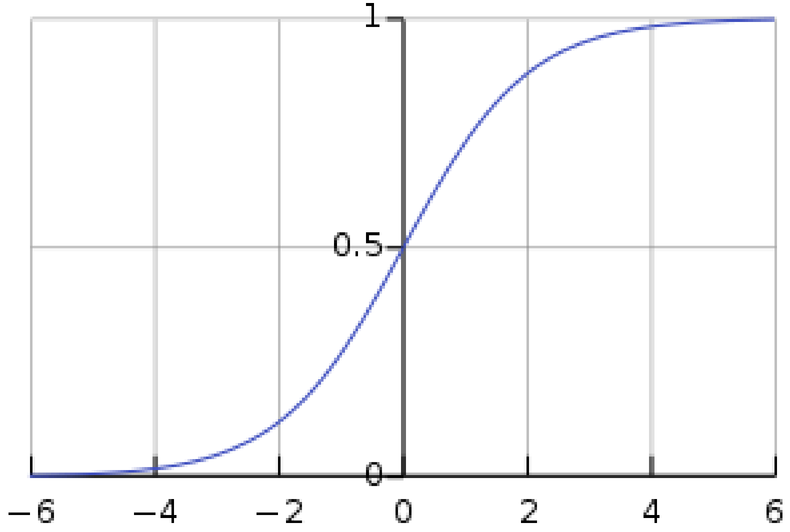
Logistic regression

The *sigmoid* function

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

or equivalently

$$\sigma(a) = \frac{e^a}{e^a + 1}$$



- $\sigma(a)$ always between 0 and 1
  - $a \to \infty, \sigma(a) \to 1$
  - $a \to -\infty, \sigma(a) \to 0$
- Monotonically increasing
- symmetry

Logistic regression

- The multivariate logistic regression model is

$$f_{\mathbf{w}}(\mathbf{x}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)$$

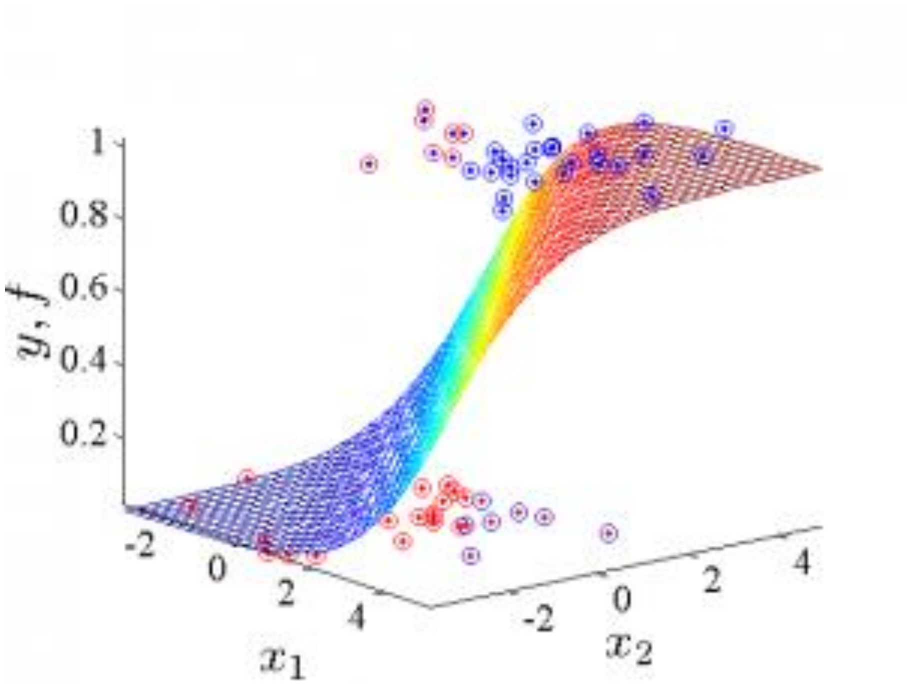
- Notice similarity to linear regression model
- However, we can *interpret* $f_{\mathbf{w}}(\mathbf{x})$ as the **probability** of $y = 1$ instead of predicting y directly
- Thresholding this probability allows us to predict the class

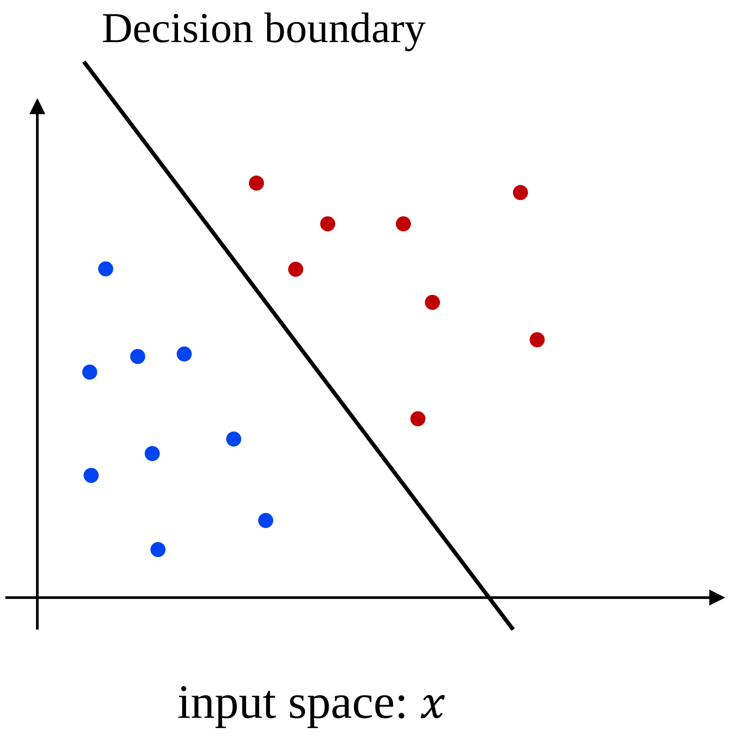
$$\hat{y} = \begin{cases} 1, & \text{if } f_{\mathbf{w}}(\mathbf{x}) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

$\sigma(0) = \frac{1}{1+e^{-0}} = 0.5 \implies$ <u>decision boundary</u> is $\mathbf{w}^{\mathrm{T}}\mathbf{x} = 0$

Logistic regression in higher dimensions is just the logistic curve **along a single direction**

The decision boundary of logistic regression is **linear**





Logistic loss

- In principle, we could still use the MSE loss:

$$\frac{1}{2n}\|\mathbf{y}-\sigma(\mathrm{X}\mathbf{w})\|_2^2$$

- However, the true output y is always 0 or 1
- A better way is to maximize **log likelihood** (more details see §5.3.2 and §5.4.3 of [1])

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\sigma(\mathbf{w}^T\mathbf{x}_i) + (1-y_i)\log\big(1-\sigma(\mathbf{w}^T\mathbf{x}_i)\big) \equiv \frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{w};\mathbf{x}_i,y_i)$$

$$\ell(\mathbf{w};\mathbf{x}_i,y_i) = \begin{cases} -\log\sigma(\mathbf{w}^T\mathbf{x}_i), & \text{if } y_i = 1 \\ -\log\big(1-\sigma(\mathbf{w}^T\mathbf{x}_i)\big), & \text{otherwise} \end{cases}$$

[1]: C.M. Bishop, and H. Bishop. *Deep learning: Foundations and Concepts*, Springer, 2023.

Regularization

<u>**Regularization**</u> is a common method to improve *generalization* by **reducing the complexity of a model** (avoiding "overfitting")

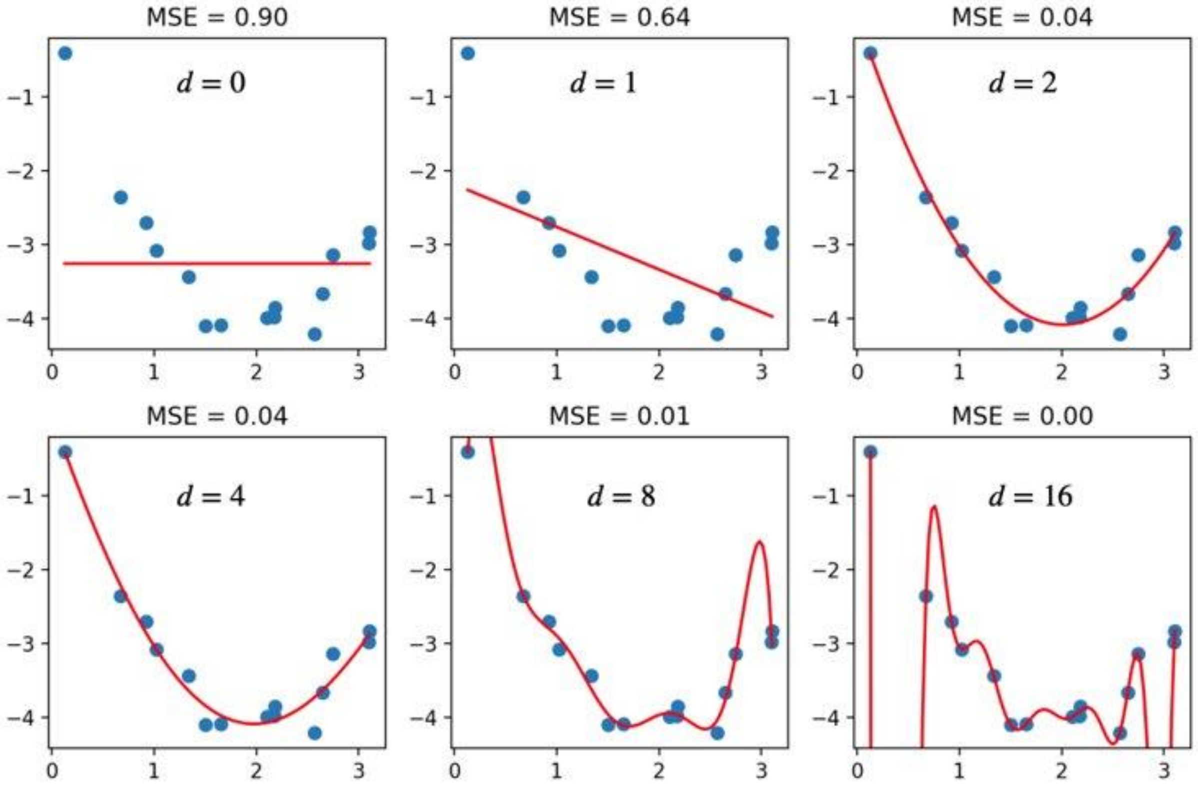
- Suppose we have 1D input data, i.e., $X \in \mathcal{R}^{n\times 1}$
- We can create pseudo polynomial features, e.g.

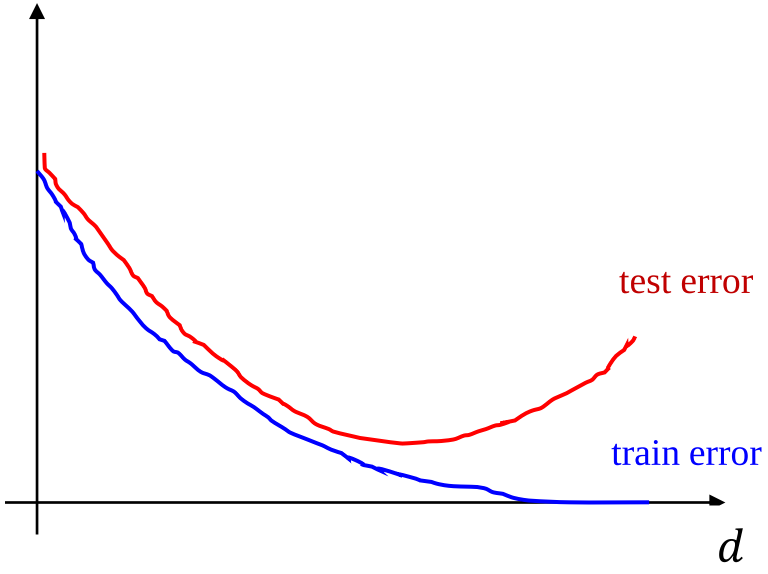
$$X' = \begin{bmatrix} x_1 & x_1^2 & x_1^3 \\ x_2 & x_2^2 & x_2^3 \\ x_3 & x_3^3 & x_3^3 \end{bmatrix} \in \mathcal{R}^{n\times 3}$$

- Linear regression can then be used to fit a polynomial model

$$y_i = b + w_1 x_i + w_2\left(x_i^2\right) + w_3\left(x_i^3\right) \ldots$$

Overfitting





Explicitly adding a regularization term:

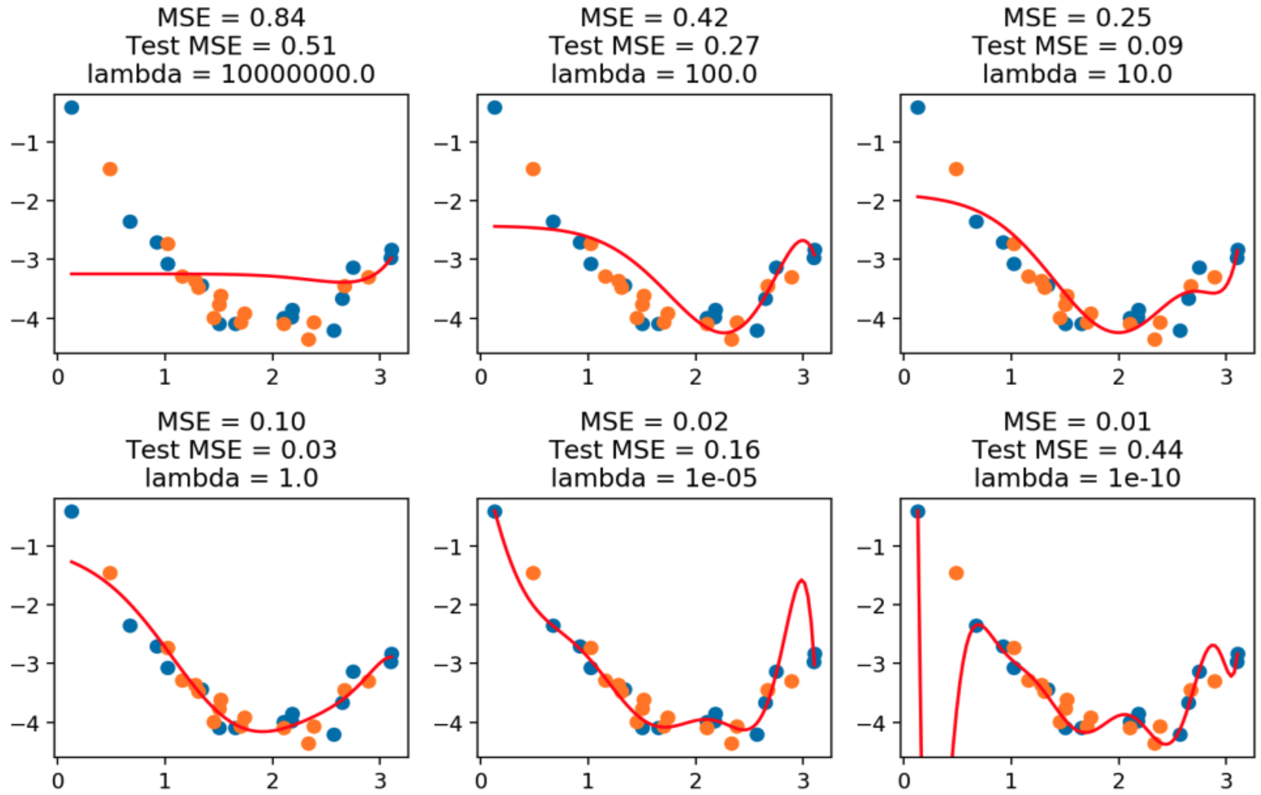
$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + \lambda \cdot P(\mathbf{w})$$

- Regularization term $P(\mathbf{w})$ independent of training data
- $P(\mathbf{w})$ non-negative
- Weight decay (L_2-regularizer) : $P(\mathbf{w}) = \|\mathbf{w}\|_2^2$
- Lasso (L_1-regularizer) : $P(\mathbf{w}) = \|\mathbf{w}\|_1^2$

Explicitly adding a <u>regularization term</u>:

- Weight decay (a.k.a. ridge regression):

$$\min_{\mathbf{w}} \|\boldsymbol{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$



Regularizing the parameters of 1D polynomial regression helps to improve test MSE if chosen appropriately.

Regularization

Explicitly adding a regularization term.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{x}_i, y_i) + \lambda \cdot P(\mathbf{w})$$

λ is a hyper-parameter:

- should be positive
- if λ is too large, penalty/regularization dominates over the loss from data, not learning well
- if λ is too small, almost no regularization