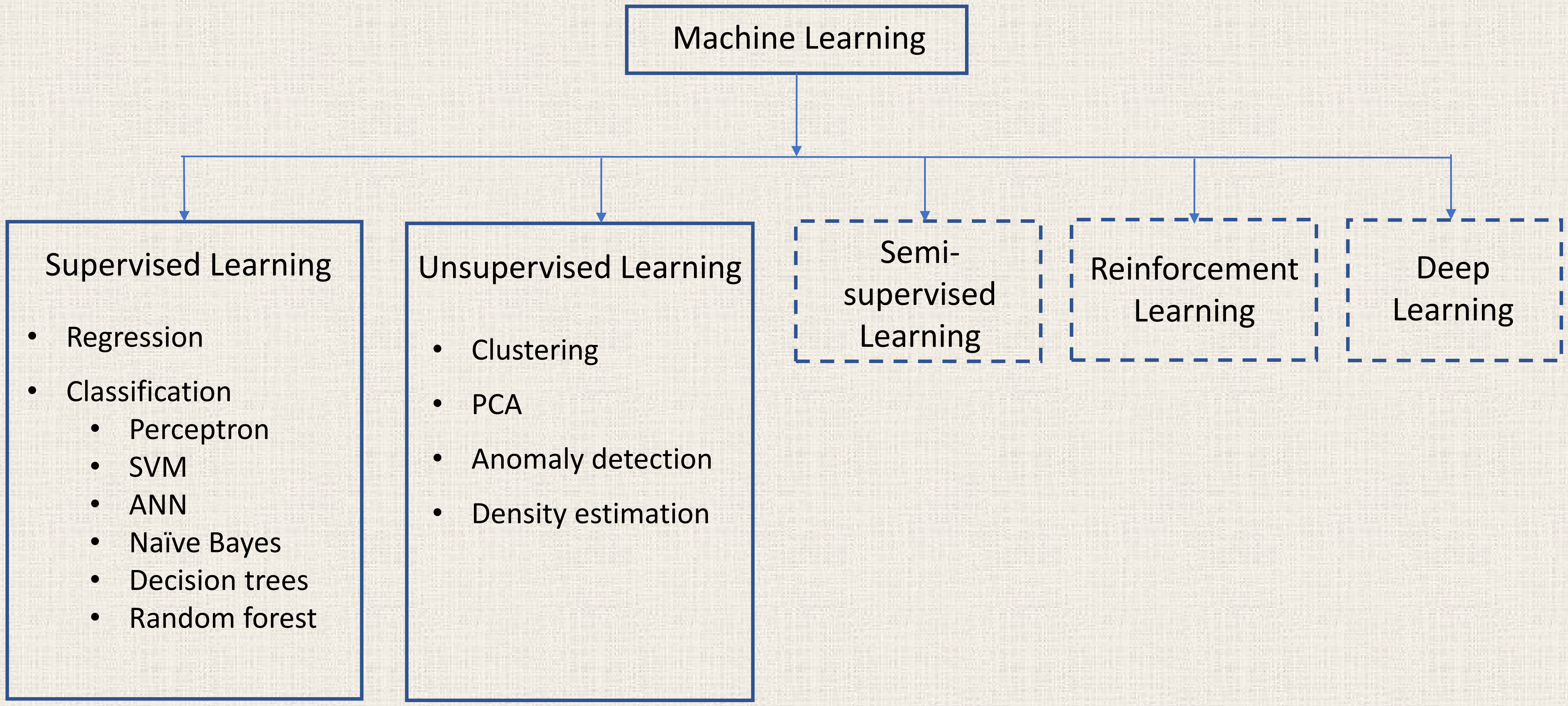
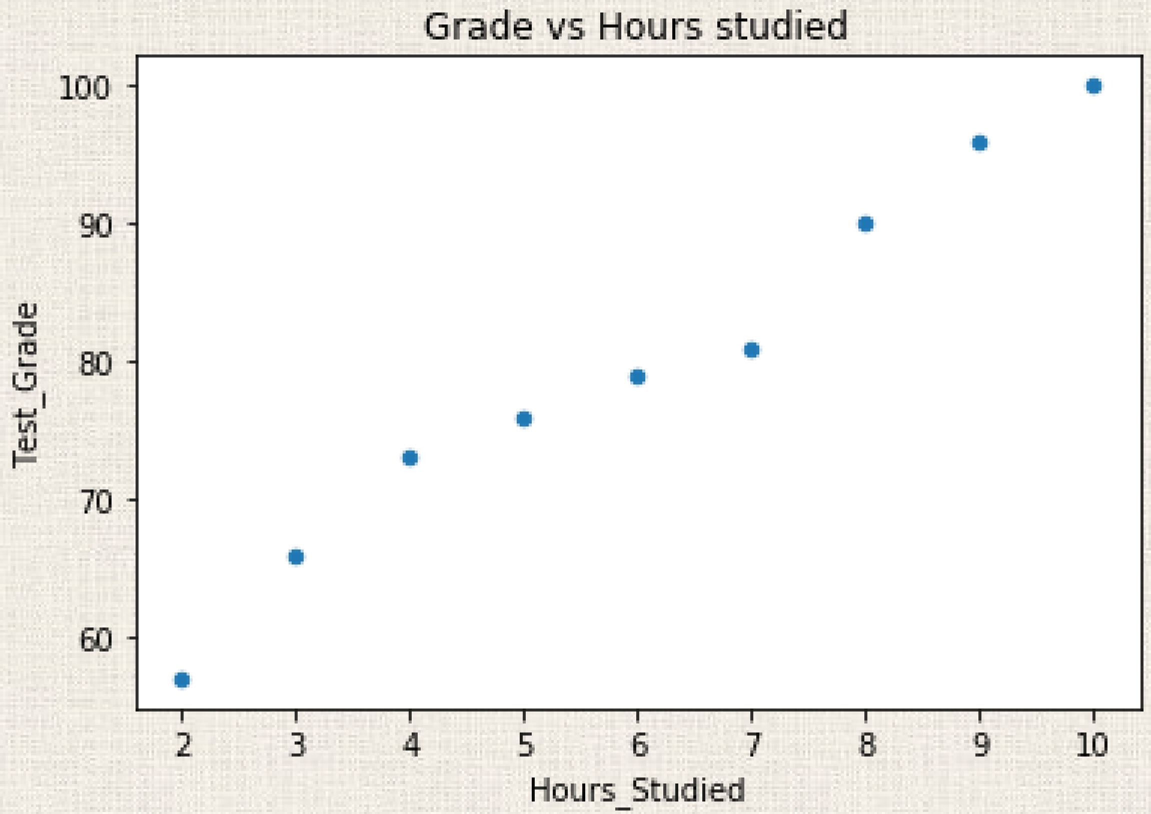
Types of Machine Learning Problems



Regression Problems

- ***Regression*** is a supervised learning problem with real valued target values $y = f(\boldsymbol{x})$
- Find the hypothesis $\hat{y} = h(\boldsymbol{x})$ to predict the target value of unseen instance $\boldsymbol{x}$

input	output
Hours_Studied	Test_Grade
2	57
3	66
4	73
5	76
6	79
7	81
8	90
9	96
10	99



House Price Prediction

output

price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea
42000	5850	3	1	two	yes	no	yes	no	no	1	no
38500	4000	2	1	one	yes	no	no	no	no	0	no
49500	3060	3	1	one	yes	no	no	no	no	0	no
60500	6650	3	1	two	yes	yes	no	no	no	0	no
61000	6360	2	1	one	yes	no	no	no	no	0	no
66000	4160	3	1	one	yes	yes	yes	no	yes	0	no
66000	3880	3	2	two	yes	no	yes	no	no	2	no
69000	4160	3	1	three	yes	no	no	no	no	0	no
83800	4800	3	1	one	yes	yes	yes	no	no	0	no

- Notice some features in the data set are categorical which need to be turned into numerical values. (https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features)

Regression Models

- ***Ordinary Least Squares*** (OLS) method, the simplest linear regression problem
- Expanded linear least squares using nonlinear ***feature mapping***
- ***Polynomial regression***
- ***Ridge regression*** (Shrinkage method for ***regularization***)
- ***Maximum Likelihood Estimate***
- ***Maximum a posterior (MAP) method***
- The relations among these methods

The ***regression problem:***

- Given the training data set $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$, with $\boldsymbol{x_i} \in \mathcal{R}^d$ the input (feature) vector, and $y_i \in \mathcal{R}$ the output (target value)

- Governed by unknown mapping $y = f(\boldsymbol{x})$

- Hypothesis set $h_{\boldsymbol{w}}(\boldsymbol{x})$ characterized by the weight vector $\boldsymbol{w} \in \mathcal{R}^d$

- Want to find $\hat{y} = h_{\boldsymbol{w}^*}(\boldsymbol{x})$ that best approximate $y = f(\boldsymbol{x})$

- We concern about the prediction accuracy, i.e., how accurate the estimated model can make prediction on unseen data

Linear hypothesis model in one-dimensional case:

- When $d = 1$, i.e., $\boldsymbol{x} = x_1$. The linear hypothesis model for this case should be

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1$$

  Where w_0 is the ***interception*** and w_1 is the ***slope***.

- The above expression can be written in the vector form as

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = [1 \;\; x_1] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \boldsymbol{x}^{\boldsymbol{T}} \boldsymbol{w}$$

  Where $\boldsymbol{x} = [1 \;\; x_1]^T, \boldsymbol{w} = [w_0 \; w_1]^T$ are the ***expended*** feature vector and ***weight*** vector.

Linear hypothesis model in multi-dimensional case:

- In general, the linear hypothesis model in high dimensional feature space is,

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

- let's expand the original feature vector and weight vector as

$$\boldsymbol{x} = [1\ x_1\ x_2\ \dots\ x_d]^T, \boldsymbol{w} = [w_0\ w_1\ w_2 \dots w_d]^T$$

- We then have,

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{T}} \boldsymbol{w} = [1\ x_1\ x_2\ \dots x_d] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \sum_{j=0}^{d} w_j x_j$$

- Since $h_{\boldsymbol{w}}(\boldsymbol{x})$ is linear with respect to $\boldsymbol{w}$ and $\boldsymbol{x}$**,** the problem is called a ***linear regression problem***.

Ordinary Least Squares (OLS) method

- The linear hypothesis model is used:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{w}$$

- For each input $\boldsymbol{x}_i$ in the data set, the predicted target value is:

$$\hat{y}_i = h_{\boldsymbol{w}}(\boldsymbol{x}_i) = \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{T}}\boldsymbol{w}\ ,\ \ i = 1,2, \dots , N.$$

  And we want that:

$$\hat{y}_i \approx y_i, \qquad i = 1,2, \dots , N$$

- We want to determine the value $\boldsymbol{w}^*$ of the weight vector so that the model prediction is a close approximation of the target value in the data set

- Let’s represent the predictions over the data set as:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \boldsymbol{Xw}$$

Where,

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}$$

The matrix $\boldsymbol{X} \in \mathcal{R}^{N\times(d+1)}$ has the input data point $\boldsymbol{x_i}$ as its i^{th} row. $\boldsymbol{X}$ is sometimes called the ***design matrix.***

Let’s specify the ***loss (cost) function***:

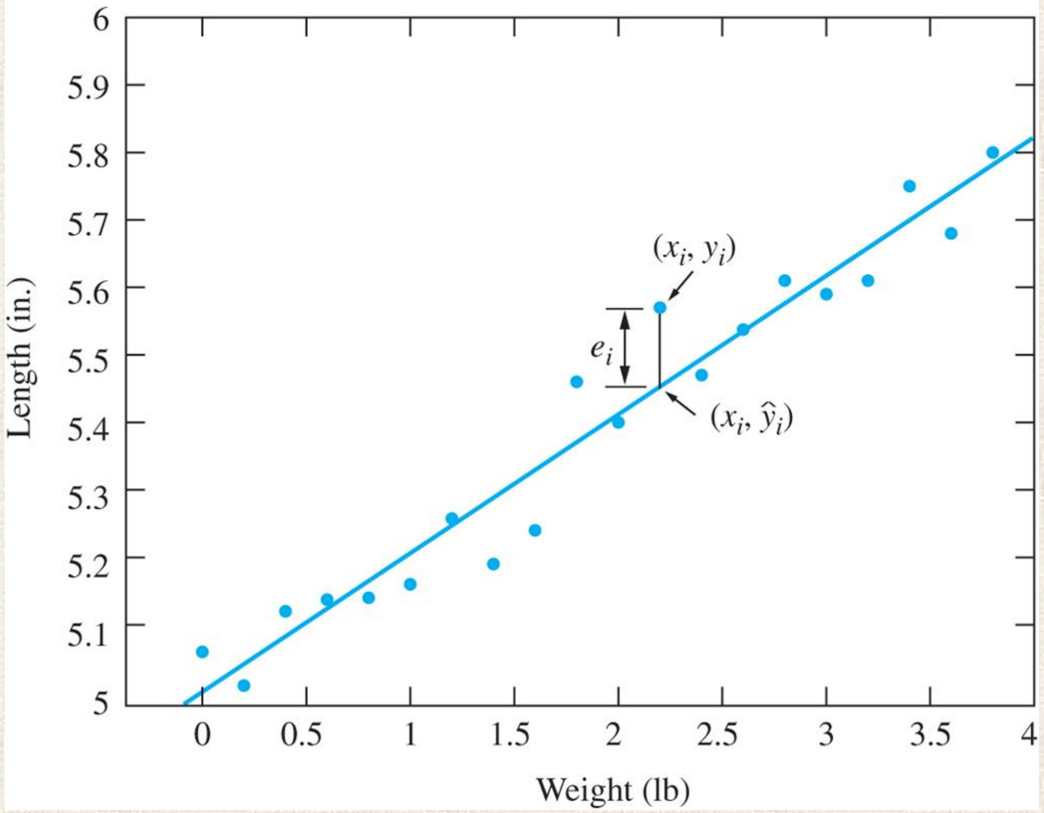
- We want to find the value of $\boldsymbol{w}$ that minimizes the **sum of squared error (SSE):**

$$L(\boldsymbol{w}) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{N} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{x}_i^T \boldsymbol{w} - y_i\right)^2 = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2{}^2$$

Where,

- $\boldsymbol{y} = [y_1, y_2, \dots, y_N]^T$ are the target values in the training data set.
- $e_i = \hat{y}_i - y_i$ is called the ***residual*** of the i_{th} instance.



- Now, we have the following ***optimization*** problem: **(Least Squares)**

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{argmin}\left\{L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2{}^2\right\}$$

  This problem can be solved to have a closed-form solution!

- Let’s solve this optimization problem:

- Let's find the gradient of the loss function $L(\boldsymbol{w})$:

$$L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2{}^2 = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$= (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w} - (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{y} - \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}$$

$$= \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} + \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}$$

  - Using the following results from matrix calculus:

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{b}\right) = \boldsymbol{b}$$

$$\nabla_{\boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x}$$

  - Then we have the gradient of $L(\boldsymbol{w})$ to be:

$$\nabla_{\boldsymbol{w}}L(\boldsymbol{w}) = \nabla_{\boldsymbol{w}}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} + \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}\right)$$

$$= \nabla_{\boldsymbol{w}}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w}\right) - 2\nabla_{\boldsymbol{w}}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}\right) + \nabla_{\boldsymbol{w}}\left(\boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}\right)$$

$$= 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

- Setting the gradient to ***0***, we have,

$$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

- if $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ is ***full rank***, then,

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{T}\boldsymbol{y} \qquad \text{(closed-form solution!)}$$

- Is $\boldsymbol{w}^*$ a global minima?

- Since the loss function is convex, $\boldsymbol{w}^*$ is the global minima .
- To show this, it is sufficed to compute the **Hessian** of L and show it is positive semi-definite:

$$\nabla^2 L(\boldsymbol{w}) = 2\boldsymbol{X}^T\boldsymbol{X}$$

$$\forall \boldsymbol{w}, \boldsymbol{w}^{\boldsymbol{T}}(2\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} = 2(\boldsymbol{X}\boldsymbol{w})^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} = 2\|\boldsymbol{X}\boldsymbol{w}\|_2{}^2 \geq 0$$

- The OLS problem has a ***closed-form*** solution:

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Notice that $\boldsymbol{X}\ and\ \boldsymbol{y}$ can be constructed using the given data set!

- Can you recognize some potential problems using this closed-form solution?
  - Computing $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$ can be very expensive when the data set is large and has many features.
  - We get a problem when $\boldsymbol{X}^T\boldsymbol{X}$ is not full rank or close to not full rank

- Do we have other method to find $\boldsymbol{w}^*$?

input	output
Hours_Studied	Test_Grade
2	57
3	66
4	73
5	76
6	79
7	81
8	90
9	96
10	99

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \\ 1 & 10 \end{bmatrix}, \ \boldsymbol{y} = \begin{bmatrix} 57 \\ 66 \\ 73 \\ 76 \\ 79 \\ 81 \\ 90 \\ 96 \\ 99 \end{bmatrix}$$

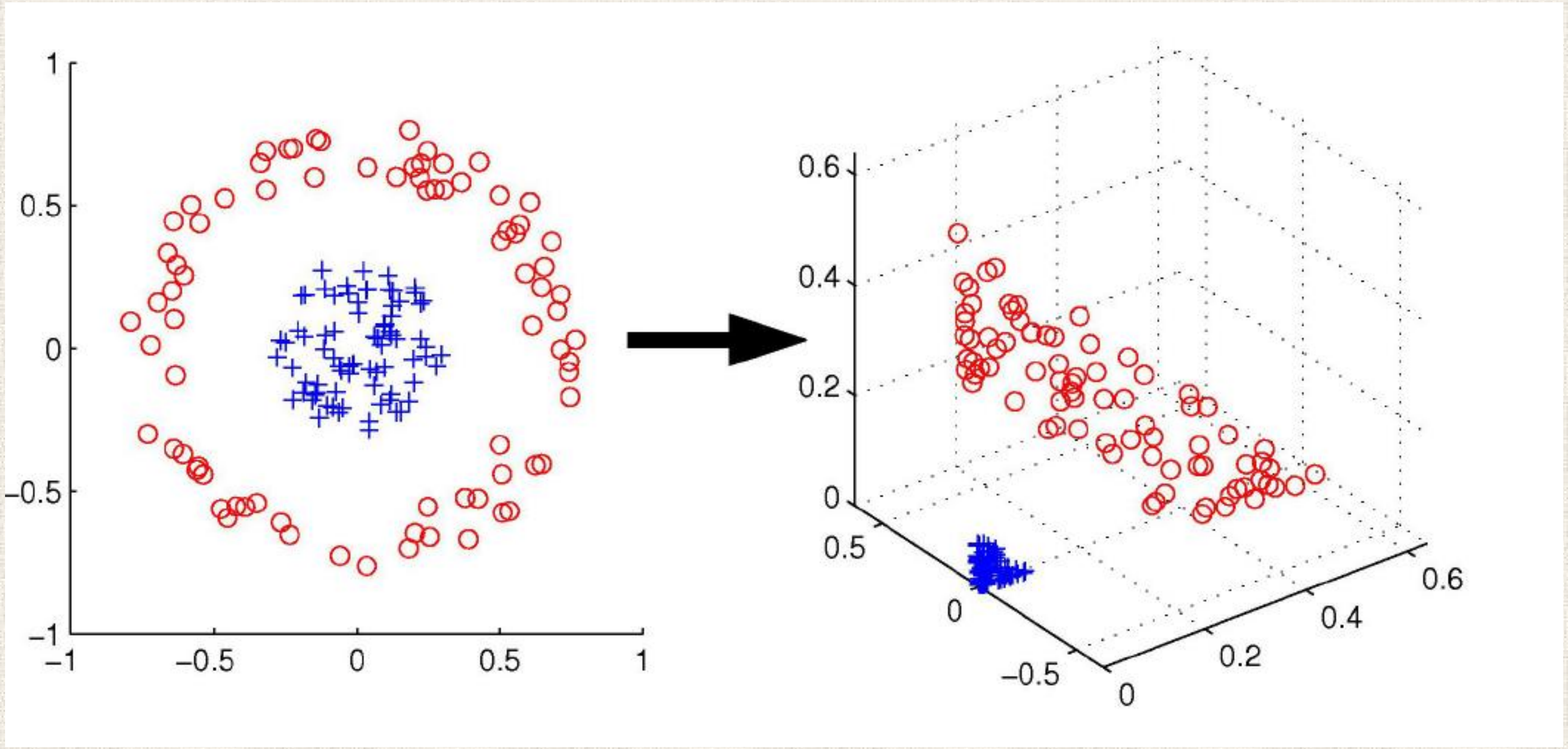
$$\boldsymbol{W}^* = \begin{bmatrix} 50 \\ 4.95 \end{bmatrix}$$

- Discuss on the linear models

  - ***Advantages:***
    - Simple, light-weighted, good ***generalization*** (especially when the training data set is small)

  - ***Disadvantages:***
    - Simple, lack of ***expressive power***, large ***bias*** for sophisticated mapping

Use Nonlinear Feature Mapping



Non-linear Feature Map

- The true input-output relationship $y = f(\boldsymbol{x})$ maybe ***nonlinear.*** It is useful to consider nonlinear model as well. This can be achieved by ***augmenting the data with nonlinear feature mapping.***

- We devise some function $\boldsymbol{\phi}: \mathcal{R}^l \to \mathcal{R}^{d+1}$, (usually $d > l$) called a ***feature map***, that maps each raw data point $\boldsymbol{x_i} \in \mathcal{R}^l$ into a vector of new features $\boldsymbol{\phi}(\boldsymbol{x_i}) \in \mathcal{R}^{d+1}$:

$$\boldsymbol{\phi}(\boldsymbol{x}_i)^T = [\phi_0(\boldsymbol{x_i}) \quad \phi_1(\boldsymbol{x_i}) \quad \cdots \quad \phi_d(\boldsymbol{x_i})]$$

Non-linear Feature Map

$$\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1l} \\ x_{21} & x_{22} & \dots & x_{2l} \\ \vdots & \vdots & \dots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nl} \end{bmatrix} \Rightarrow \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_d(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_d(\boldsymbol{x}_2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_d(\boldsymbol{x}_N) \end{bmatrix}$$

Expanded Linear Least Squares

- The hypothesis function then becomes:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=0}^{d} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

(linear combination of nonlinear mapping functions)

- Note that this model is still *linear* with respect to the mapped features, but it is nonlinear with respect to the original data if $\phi_j s$ are nonlinear. This model is ***linear with respect to the weights!***.

- The component functions ϕ_j are sometimes called ***basis functions.*** (polynomial, piecewise linear, Gaussian, sigmoid, etc.)

- Example ***basis functions***:

$$\phi_j(\boldsymbol{x}) = e^{-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_j)^2}{2s^2}} \quad (Gaussian)$$

$$\phi_j(\boldsymbol{x}) = \frac{1}{1 + e^{-\frac{\boldsymbol{x}-\boldsymbol{\mu}_j}{s}}} \quad (sigmoid)$$

- we can then use least squares to estimate the weights $\boldsymbol{w}$ to minimize $||\boldsymbol{\Phi w} - \boldsymbol{y}||^2$, where, $\boldsymbol{\Phi} \in \mathcal{R}^{N\times(d+1)}$ is the ***design matrix*** as follows:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_d(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_d(\boldsymbol{x}_2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_d(\boldsymbol{x}_N) \end{bmatrix}$$

- The solution of the ***expanded least square problem*** is:

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$$

Polynomial Feature Map (Transformation)

- ***Polynomial feature mapping*** for ***one dimensional*** input:

$$x \Rightarrow \boldsymbol{\phi}(x) = [1 \quad x \quad x^2 \quad x^3 \quad \cdots \quad x^m]^T$$

- The hypothesis models set becomes:

$$h_w(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$$

$$= [1 \quad x \quad x^2 \quad x^3 \quad \cdots \quad x^m] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

$$= \boldsymbol{\phi}(x)^T \boldsymbol{w}$$

- The ***univariate*** polynomial model with degree m:

$$y = w_0 + w_1 x \qquad \text{(m=1)}$$

$$y = w_0 + w_1 x + w_2 x^2 \qquad \text{(m=2)}$$

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \qquad \text{(m=3)}$$

$$\vdots$$

$$y = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$$

- ***Bivariate polynomial model*** (two-dimensional input, m = 2):

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_{11} x_1^2 + w_{22} x_2^2 + w_{12} x_1 x_2$$

Polynomial Features

- A big reason that we care about polynomial features is that ***any smooth function can be approximated arbitrarily closely by some polynomial***.

- For this reason, polynomial are said to be a ***universal approximator.***

- For simplicity, we use a 1D input as an example to find the solution.
- The data set: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ governed by the unknown relation: $y = f(x)$
- The hypothesis model:

$$\hat{y} = h_{\boldsymbol{w}}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m = \boldsymbol{\phi}(x)^T \boldsymbol{w}$$

Where, $\boldsymbol{w} = [w_0 \quad w_1 \quad w_2 \quad \cdots \quad w_m]^T$ is the weight vector, and

$\boldsymbol{\phi}(x)^T = [1 \quad x \quad x^2 \quad x^3 \quad \cdots \quad x^m]^T$ is the mapped feature vector.

- This model is still linear with respect to the weight vector $\boldsymbol{w}$

- From the linear hypothesis model, we have,

$$\hat{y}_1 = w_0 + w_1 x_1 + w_2 x_1^2 + \cdots + w_m x_1^m$$
$$\hat{y}_2 = w_0 + w_1 x_2 + w_2 x_2^2 + \cdots + w_m x_2^m$$
$$\vdots$$
$$\hat{y}_N = w_0 + w_1 x_N + w_2 x_N^2 + \cdots + w_m x_N^m$$

In matrix form,

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

i.e.,

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}$$

- To minimize the ***sum of squared error***:

$$L(\boldsymbol{w}) = {\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_{\boldsymbol{2}}}^2$$

- We have,

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Note that $m < N$, i.e., the number of data is greater than the number of features.

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix}$$

is the ***design matrix***.

- Construction of the ***design matrix*** $\boldsymbol{X}$:

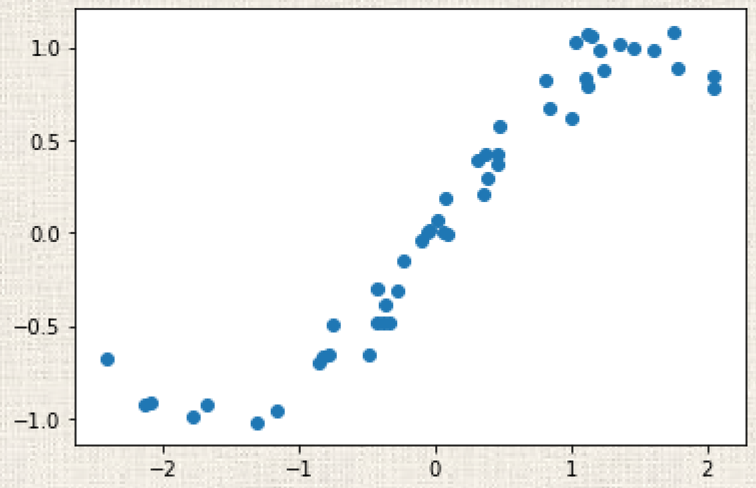
$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{bmatrix}$$

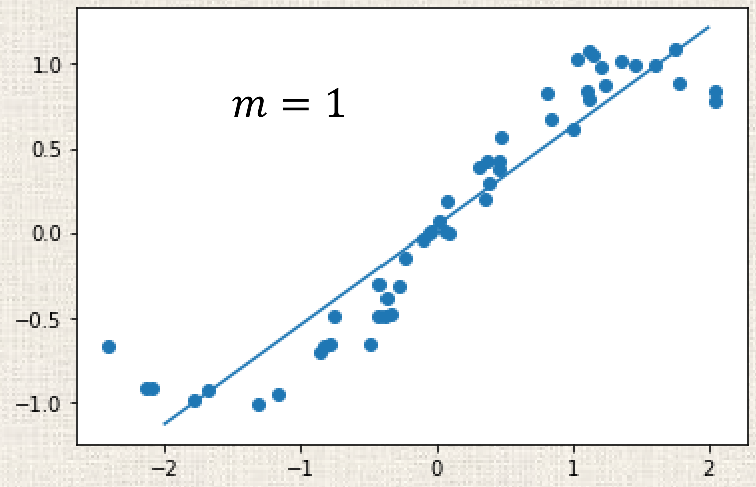
Hours_Studied	Test_Grade
2	57
3	66
4	73
5	76
6	79
7	81
8	90
9	96
10	100

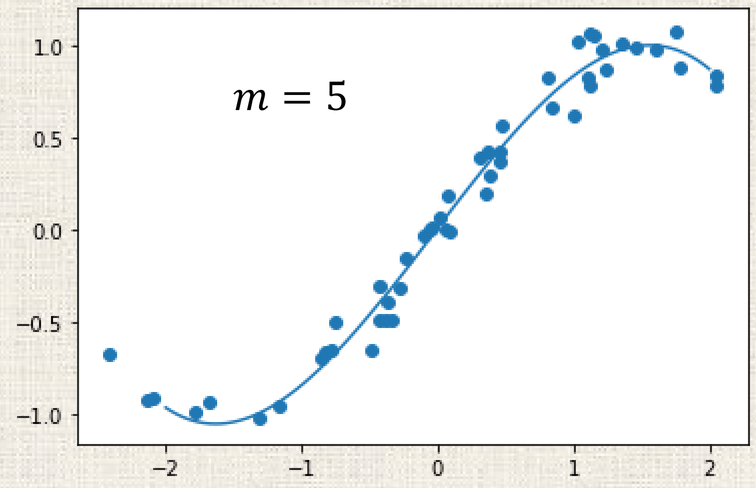
When $m = 2$

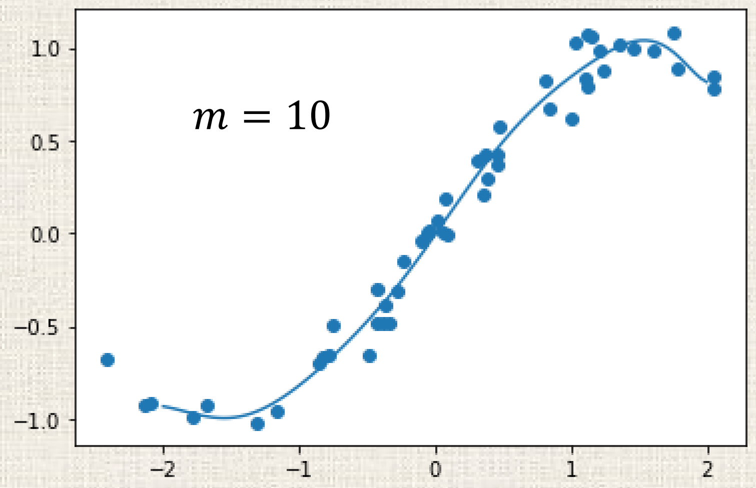
$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{bmatrix}$$

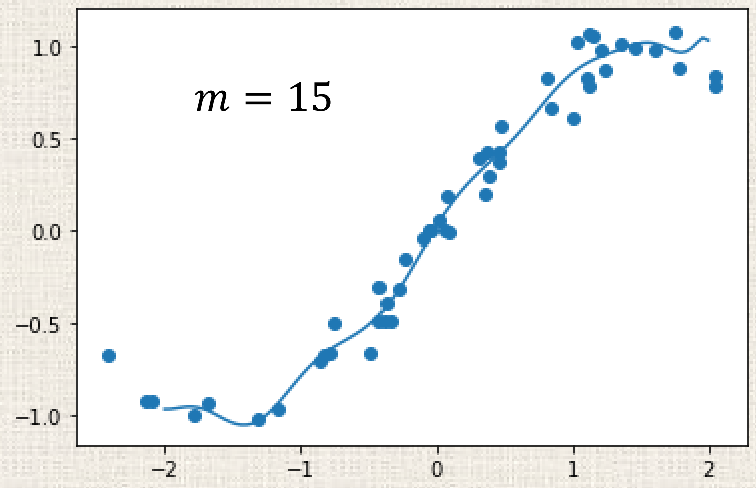
Polynomial Regression Example

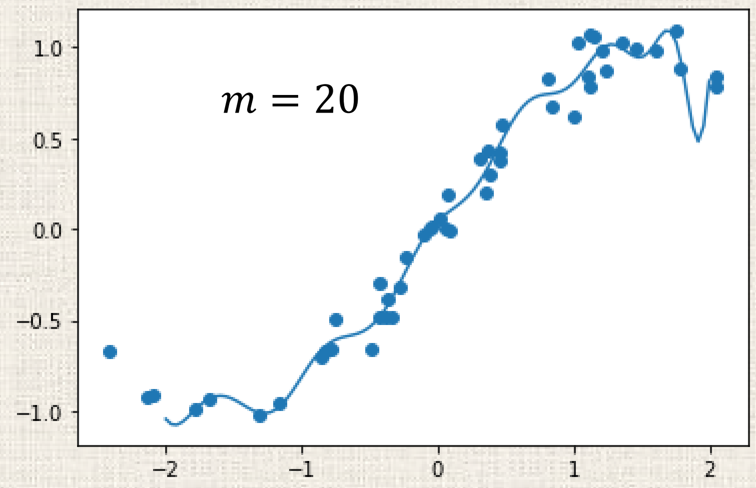












Regularization

- ***Numerical instability*** with the closed-form solution formula:

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

  Numerical instability arise when the features of the data are close to ***collinear***, i.e., some features are linear combinations of others, causing the input matrix $\boldsymbol{X}$ to lose its rank or have singular values that very close to 0

  This will cause $\boldsymbol{X}^T\boldsymbol{X}$ to be not invertible or it is invertible but $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$ will be huge, and the variance of the estimated coefficients $\boldsymbol{w}^*$ will be enormous.

- ***Overfitting happens when complex hypothesis models are used (****some components of the weight vector becomes extremely large!****)***

- ***Bad generalization*** (bad prediction when applying the estimated model to unseen examples)

Ridge Regression (a shrinkage method)

- There is a very simple solution to these issues: *penalize the large entries of* $\boldsymbol{w}$ *to prevent then from becoming too large*.

- We can do this by adding a penalty term constraining the norm of $\boldsymbol{w}$. **-- regularization**

- For a fixed small scalar $\lambda > 0$, we now have,

$$L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_{\boldsymbol{2}}{}^{2} + \lambda\|\boldsymbol{w}\|_{2}{}^{2}$$

  - λ is called a ***hyper parameter*** that measures the sensitivity to the values in $\boldsymbol{w}$.
  - λ is a value that we choose through ***validation***.

Ridge Regression

- L2 regularization:

$$L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_{\boldsymbol{2}}{}^2 + \lambda\|\boldsymbol{w}\|_{\boldsymbol{2}}{}^2$$

$$= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} + \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y} + \lambda\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$$

- Take the gradient of $L(\boldsymbol{w})$ and let it be 0, we have,

$$\nabla_{\boldsymbol{w}}L(\boldsymbol{w}) = 0$$

$$\Rightarrow 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} + 2\lambda\boldsymbol{w} = 0$$

$$\Rightarrow \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)\boldsymbol{w} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

$$\Rightarrow \boldsymbol{w}_{ridge} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

- This value is guaranteed to achieve the unique global minimum, because the objective function is convex.

Ridge Regression

- Let's find the ***Hessian*** of $L(w)$:

$$\nabla^2 L(\boldsymbol{w}) = 2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + 2\lambda\boldsymbol{I}$$

$$\forall \boldsymbol{w} \neq 0, \boldsymbol{w}^{\boldsymbol{T}}\left(2\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + 2\lambda\boldsymbol{I}\right)\boldsymbol{w} = (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w} + \lambda\boldsymbol{w}^T\boldsymbol{w} = \|\boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2 > 0$$

i.e., $\nabla^2 L(\boldsymbol{w})$ is ***positive definite*** (*PD*). The minimum is unique!

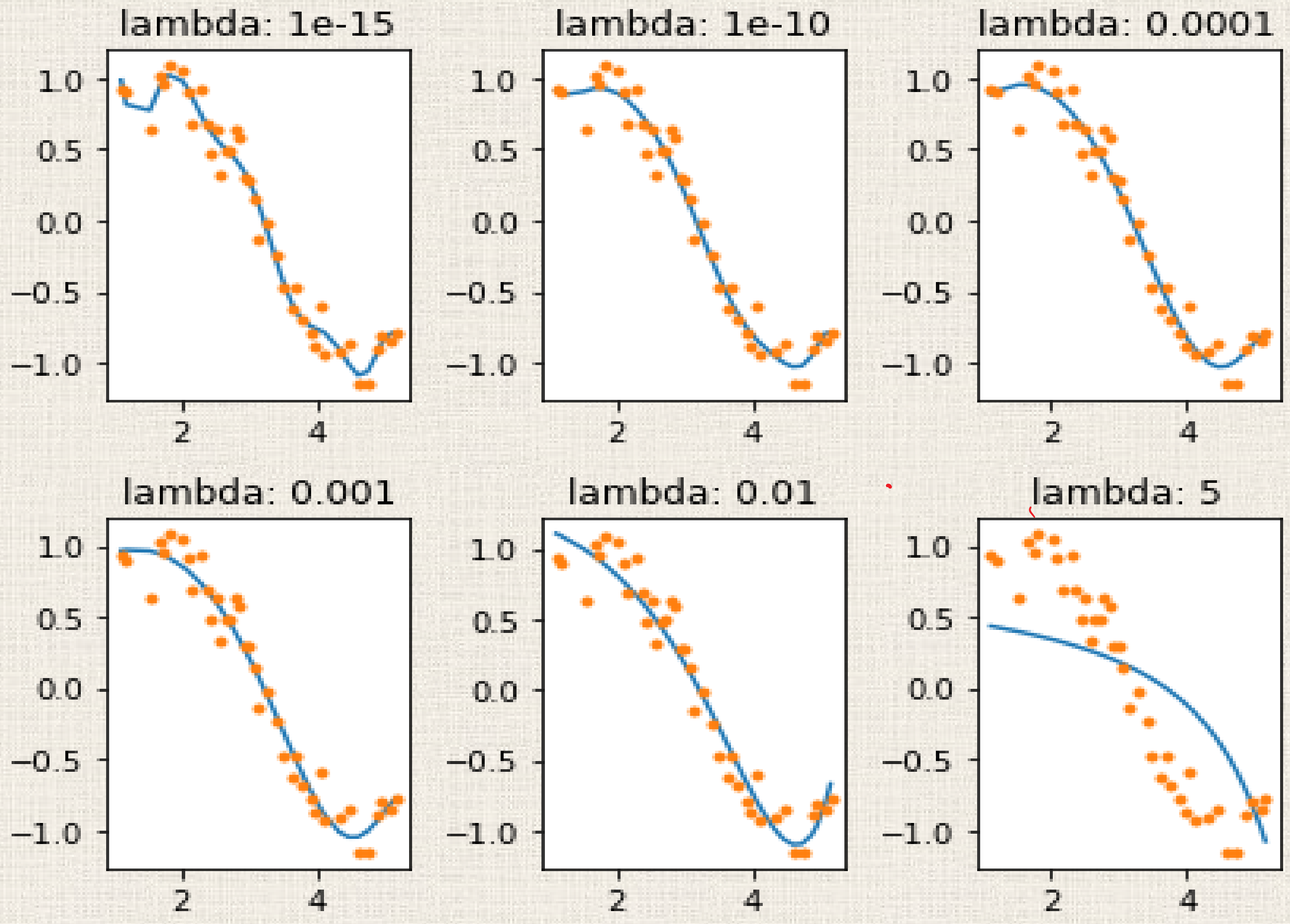
- Now with our slight tweak, the matrix $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \lambda\boldsymbol{I}$ has become full rank and invertible, meaning the numerical instability is solved.

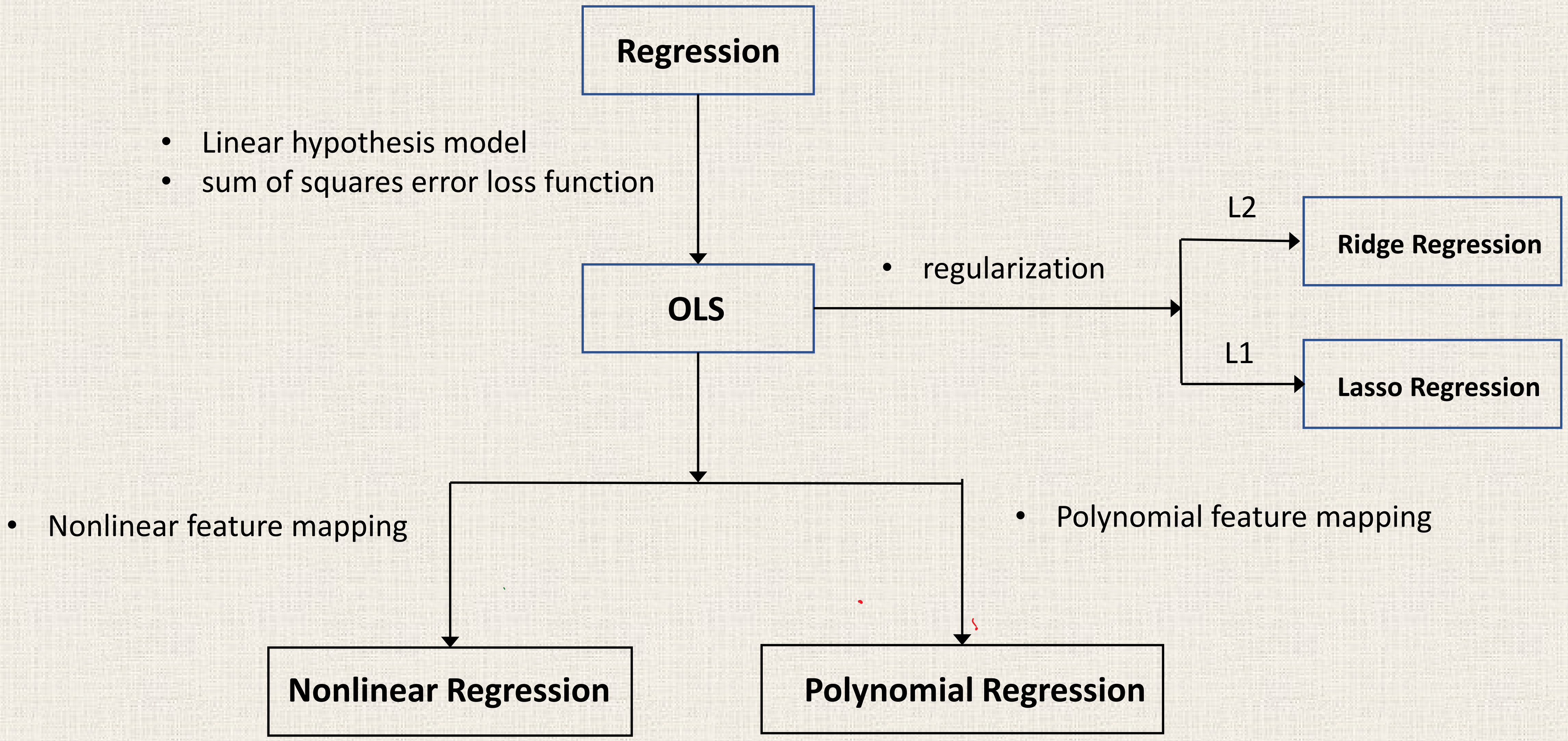
Lasso Regression (L1 regularization)

$$\boldsymbol{w}_{Lasso} = \underset{\boldsymbol{w}}{argmin} \left\{ \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_{\mathbf{2}}{}^{2} + \lambda \sum_{j=0}^{d} |w_j| \right\}$$

Regularization technique can be applied to overcome overfitting!!!

$$L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2{}^2 + \lambda\|\boldsymbol{w}\|_2{}^2$$




Regression
Linear hypothesis model
sum of squares error loss function
OLS
regularization
L2
Ridge Regression
L1
Lasso Regression
Nonlinear feature mapping
Polynomial feature mapping
Nonlinear Regression
Polynomial Regression

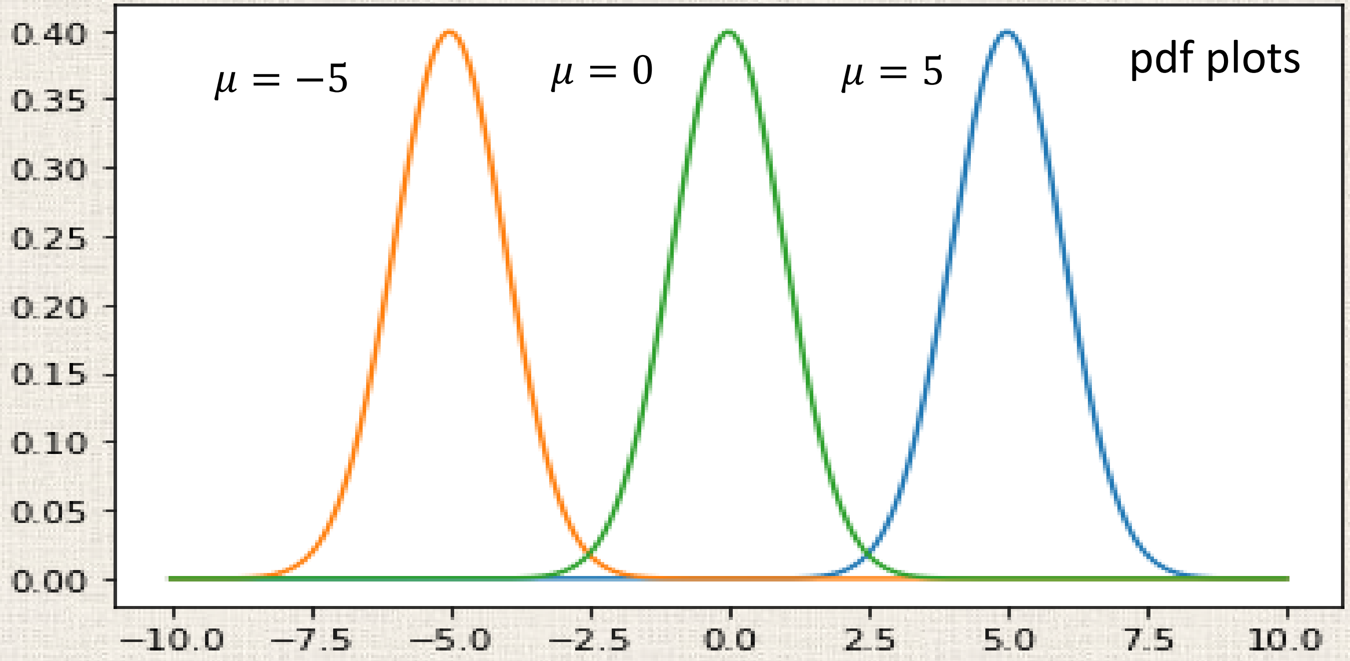
The models developed so far do not involve any assumptions on distributions of data

The Maximum Likelihood Estimate (MLE) Model

- Let random variable $X \sim N(\mu, 1)$. To estimate the value of μ, a random sample of size 5 was drawn from X as follows:

0.1	**-0.9**	**0.8**	**0.2**	**-0.1**

- Let's also assume that μ can only take three possible values, $\mu = -5,\ 0,\ 5$. Based on the sampled data, which is the most likely value of μ?



- Let random variable $X \sim N(\mu, 1)$. To estimate the value of μ, a random sample of size 5 was drawn from X as follows:

0.1	**-0.9**	**0.8**	**0.2**	**-0.1**

- Let's define the following likelihood function:

$$L(\mu) = p\big((x_1 = 0.1) \cap (x_2 = -0.9) \cap (x_3 = 0.8) \cap (x_4 = 0.2) \cap (x_5 = -0.1)\big)$$

$$= p(x_1 = 0.1|\mu)p(x_2 = -0.9|\mu)p(x_3 = 0.8|\mu)p(x_4 = 0.2|\mu)p(x_5 = -0.1|\mu)$$

$$= \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(0.1-\mu)^2}{2}}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(-0.9-\mu)^2}{2}}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(0.8-\mu)^2}{2}}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(0.2-\mu)^2}{2}}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(-0.1-\mu)^2}{2}}\right)$$

- Maximizing ***likelihood function*** is equivalent to maximizing the ***log likelihood,*** which is:

$$\ln(L(\mu)) = \ln\left(\frac{1}{\sqrt{2\pi}}e^{\frac{-(0.1-\mu)^2}{2}}\right) + \ln\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{(-0.9-\mu)^2}{2}}\right) + \ln\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{(0.8-\mu)^2}{2}}\right) + \ln\left(\frac{1}{\sqrt{2\pi}}e^{\frac{-(0.2-\mu)^2}{2}}\right) + \ln\left(\frac{1}{\sqrt{2\pi}}e^{\frac{-(-0.1-\mu)^2}{2}}\right)$$

$$= -\frac{5}{2}\ln(2\pi) - \frac{(0.1-\mu)^2}{2} - \frac{(-0.9-\mu)^2}{2} - \frac{(0.8-\mu)^2}{2} - \frac{(0.2-\mu)^2}{2} - \frac{(-0.1-\mu)^2}{2}$$

- Take the derivative of the ***log likelihood function*** w.r.t μ and let it equal to 0:

$$(0.1-\mu) + (-0.9-\mu) + (0.8-\mu) + (0.2-\mu) + (-0.1-\mu) = 0$$

$$\Rightarrow -0.1 - 5\mu = 0$$

$$\Rightarrow \hat{\mu} = -0.02$$

Maximum Likelihood Estimate (MLE) – Probability Models

- In supervised learning, we assume that there exists a true underlying model mapping inputs to outputs: $f(\boldsymbol{x})$

- The only information we have about the true model is via a data set:

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$$

Where $\boldsymbol{x}_i \in \mathcal{R}^d$ is the input and $y_i \in \mathcal{R}$ is the observation of a random variable Y_i, i.e.,

$$Y_i = h_{\boldsymbol{w}}(\boldsymbol{x}_i) + Z_i$$

- We assume that $\boldsymbol{x}_i$ is a fixed value, while Z_i is a ***random variable***.
- In most contexts, we assume that $Z_i \sim N(0, \sigma^2), i = 1,2, \ldots N$. i.e., Z_i are ***independent identically distributed*** (***i.i.d.) zero mean Gaussians***. Then,

$$Y_i \sim N(h_{\boldsymbol{w}}(\boldsymbol{x}_i), \sigma^2)$$

- ***Maximum Likelihood Estimate (MLE)***, the goal is to find the hypothesis model that maximizes the ***likelihood function*** of the data, i.e.,

$$\widehat{\boldsymbol{w}}_{MLE} = \underset{\boldsymbol{w}}{argmax}\, L(\boldsymbol{w}; \mathcal{D})$$

Where, the ***likelihood function*** is:

$$L(\boldsymbol{w}, \mathcal{D}) = p(Y_1 = y_1, \dots, Y_N = y_n | \boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{w}) = p(y_1, \dots, y_N | \boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{w})$$

$$= p(y_1|\boldsymbol{x}_1, \boldsymbol{w}) p(y_2|\boldsymbol{x}_2, \boldsymbol{w}) \cdots p(y_N|\boldsymbol{x}_N, \boldsymbol{w}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

- Maximizing ***likelihood function*** is equivalent to maximizing the ***log likelihood,*** which is:

$$l(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = ln\big(L(\boldsymbol{w}; \mathcal{D})\big) = \sum_{i=1}^{N} ln(p(y_i|\boldsymbol{x}_i, \boldsymbol{w}))$$

- When $Y_i \sim N(h_{\boldsymbol{w}}(\boldsymbol{x}_i), \sigma^2)$, we have, the ***probability density function (pdf)*** of Y_i is:

$$p(Y_i = y_i | \boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{\left(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i)\right)^2}{\sigma^2}}$$

- Then, the ***nature log likelihood*** becomes:

$$l(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = -\sum_{i=1}^{N} \frac{\left(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i)\right)^2}{2\sigma^2} - N\, ln\left(\sqrt{2\pi}\sigma\right)$$

- Maximizing $l(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$ is equivalent to the following:

$$\widehat{\boldsymbol{w}}_{MLE} = \underset{\boldsymbol{w}}{argmin} \left[\sum_{i=1}^{N} \frac{\left(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i)\right)^2}{2\sigma^2} + N\, ln\left(\sqrt{2\pi}\sigma\right)\right] = \underset{\boldsymbol{w}}{argmin} \left[\sum_{i=1}^{N} \left(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i)\right)^2\right]$$

- In the linear regression problem, our hypothesis model has the form:

$$h_{\boldsymbol{w}}(x_i) = \boldsymbol{x}_i^T \boldsymbol{w}$$

Then, the problem becomes:

$$\widehat{\boldsymbol{w}}_{MLE} = \underset{\boldsymbol{w}}{argmin} \left[\sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 \right] = \underset{\boldsymbol{w}}{argmin} \{ \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2{}^2 \}$$

This is just the ***OLS*** problem!

- This means that ***MLE*** is a probabilistic justification for why ***sum of squared error*** (**SSE**) is a good metric for evaluating a regression model!!!

- ***Question:*** what are the differences between ***MLE*** model and ***OLS*** model of regression?
  - OLS is a special case of MLE when Gaussian distribution is assumed for Y_i and linear hypothesis model is used
  - Other assumption can also be used for MLE!

Bayes’ Theorem

- ***Bayesian reasoning*** provides the basis for learning algorithms that directly manipulate probabilities.
- ***Bayes’ Theorem***:

  Given a ***hypothesis*** h and a piece of **evidence** E, *Bayes Theorem* states that

$$P[h|E] = \frac{P[E|h]P[h]}{P[E]}$$

  Where,

  - $P[h|E]$ is the ***posterior probability***,
  - $P[h]$ is the ***prior probability***,
  - $P[E|h]$ is called ***likelihood***.
- ***Bayes’ theorem*** describes how to update the ***probabilities of hypothesis when given an evidence***.

- ***Bayes' Theorem***:

Given multiple ***hypothesis*** $h_i, i = 1,2,\dots M$ that are mutual exclusive and exhaustive, and a piece of **evidence** E, *Bayes Theorem* states that

$$P[h_i|E] = \frac{P[E|h_i]P[h_i]}{\sum_{k=1}^{M} P[E|h_k]P[h_k]}$$

- ***Bayes' theorem*** describes how to update the ***probabilities of hypothesis when given an evidence***.

An example on Bayes’ Theorem applied on diagnosis problem

Consider a simple medical diagnosis problem as follows:

- Two alternative hypothesis:
    - h: the patient has a particular form of disease;
    - h^c: the patient does not have the disease;
- We have prior knowledge that over the entire population of people only 0.8% have this disease
- The available data is from a particular lab test with two possible outcomes: *positive* or *negative*
- Furthermore, the lab test is only an imperfect indicator of the disease
    - The test returns a correct positive result in only 99% of the cases where the disease is present
    - The test returns a correct negative result in only 99.5% of the cases in which the disease is not present

- Define the following events:
  - h: a person has the disease; h^c: a person does not have the disease;
  - $\oplus$: the lab test result is positive; $\ominus$: the lab test result is negative
- Then, the following probabilities are given:

$$P(h) = 0.008; \quad P(h^c) = 0.992 \ (\textbf{\textit{prior}})$$

$$P(\oplus\,|h) = 0.99 \Rightarrow P(\ominus\,|h) = 0.01 \ (\textbf{\textit{false negative rate}})$$

$$P(\ominus\,|h^c) = 0.995 \Rightarrow P(\oplus\,|h^c) = 0.005 \ (\textbf{\textit{false positive rate}})$$

- Suppose we now have a patient for whom the lab test returns a positive result. Should we diagnose the patient as having the disease or not?
- Using the ***Bayes' Theorem***, we can find the posterior probability of $h \; and \; h^c$:

$$P(h|\oplus) = \frac{P(\oplus|h)P(h)}{P(\oplus)} = \frac{P(\oplus|h)P(h)}{P(\oplus|h)P(h) + P(\oplus|h^c)P(h^c)}$$

$$= \frac{0.99 \times 0.008}{0.99 \times 0.008 + 0.005 \times 0.992} = 0.6149$$

$$P(h^c|\oplus) = \frac{P(\oplus|h^c)P(h^c)}{P(\oplus)} = \frac{P(\oplus|h^c)P(h^c)}{P(\oplus|h)P(h) + P(\oplus|h^c)P(h^c)}$$

$$= \frac{0.003 \times 0.992}{0.99 \times 0.008 + 0.005 \times 0.992} = 0.3851$$

- Since $P(h|\oplus) > P(h^c|\oplus)$, we have $h^* = h$, i.e., we should diagnose the patient as ***having the disease***.

- Alternatively, ratio of the posterior probabilities can be calculated as:

$$\frac{P(h|\oplus)}{P(h^c|\oplus)} = \frac{P(\oplus|h)P(h)}{P(\oplus|h^c)P(h^c)} = \frac{0.99 \times 0.008}{0.005 \times 0.992} = 1.6$$

- What is your conclusion based on this ratio of posterior probabilities?

Maximum a Posterior (MAP) method – A Probability model

- ***In many learning scenarios, the learner considers some set of candidate hypothesis H and is interested in finding the most probable hypothesis $h \in H$ given the observed data $\mathcal{D}$.***
- Any such maximally probable hypothesis is called a ***maximum a posterior (MAP) hypothesis***.

- Under this setting, applying ***Bayes' Theorem*** gives:

$$h_{MAP} = \underset{h \in H}{argmax}\, P[h|\mathcal{D}]$$

$$= \underset{h \in H}{argmax} \left\{ \frac{P[\mathcal{D}|h]P[h]}{P[\mathcal{D}]} \right\} = \underset{h \in H}{argmax} \{P[\mathcal{D}|h]P[h]\}$$

- Notice that in the last step, we dropped $P[\mathcal{D}]$ because it is a constant independent of h.

- When the hypothesis models are parameterized by the weight vector $\boldsymbol{w}$, the problem becomes:

$$\widehat{\boldsymbol{w}}_{MAP} = \underset{\boldsymbol{w}}{argmax}\{P[h_{\boldsymbol{w}}|\mathcal{D}]\} = \underset{\boldsymbol{w}}{argmax}\{P[\mathcal{D}|h_{\boldsymbol{w}}]P[h_{\boldsymbol{w}}]\}$$

$$= \underset{\boldsymbol{w}}{argmax}\{ln\, P[\mathcal{D}|h_{\boldsymbol{w}}] + ln\, P[h_{\boldsymbol{w}}]\}$$

$$= \underset{\boldsymbol{w}}{argmin}\{-ln\, P[\mathcal{D}|h_{\boldsymbol{w}}] - ln\, P[h_{\boldsymbol{w}}]\}$$

Where $P[\mathcal{D}|h_{\boldsymbol{w}}]$ is the ***likelihood of data***. $P[h_{\boldsymbol{w}}]$ is the ***prior*** over the hypothesis model

- Assuming that every hypothesis in H is ***equally likely***, i.e., every $h \in H$ has *equal prior probability, i.e.,*

$$P[h_{\boldsymbol{w}}] = constant$$

The ***MAP hypothesis*** becomes the ***maximum likelihood (ML)*** hypothesis:

$$h_{MAP} = \underset{h \in H}{argmax}\, P[\mathcal{D}|h] = h_{ML}$$

- Hence, we have,

$$\hat{\boldsymbol{w}}_{MAP} = \underset{\boldsymbol{w}}{argmin}\left\{-\sum_{i=1}^{N} ln[p(y_i|\boldsymbol{x}_i,\boldsymbol{w})] - ln[p(\boldsymbol{w})]\right\}$$

- Let's assume that $Y_i \sim N(h_{\boldsymbol{w}}(\boldsymbol{x}_i), \sigma^2)$, and for the prior $p(\boldsymbol{w})$, we assume that the components w_j are ***i.i.d Gaussian***, i.e., $w_j \sim N(w_{j0}, \sigma_h^2)$. Then, we have,

$$\hat{\boldsymbol{w}}_{MAP} = \underset{\boldsymbol{w}}{argmin}\left\{\frac{\sum_{i=1}^{N}(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))^2}{2\sigma^2} + \frac{\sum_{j=1}^{d}(w_j - w_{j0})^2}{2\sigma_h^2}\right\}$$

$$= \underset{\boldsymbol{w}}{argmin}\left\{\sum_{i=1}^{N}(y_i - h_w(\boldsymbol{x}_i))^2 + \frac{\sigma^2}{\sigma_h^2}\left(\sum_{j=1}^{d}(w_j - w_{j0})^2\right)\right\}$$

- In the linear regression problem, we have $h_{\boldsymbol{w}}(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{w}$. When $w_{j0} = 0$, the problem becomes:

$$\hat{\boldsymbol{w}}_{MAP} = \underset{\boldsymbol{w}}{argmin} \left\{ \sum_{i=1}^{N} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w} \right)^2 + \frac{\sigma^2}{\sigma_h^2} \left(\sum_{j=1}^{d} w_j^2 \right) \right\}$$

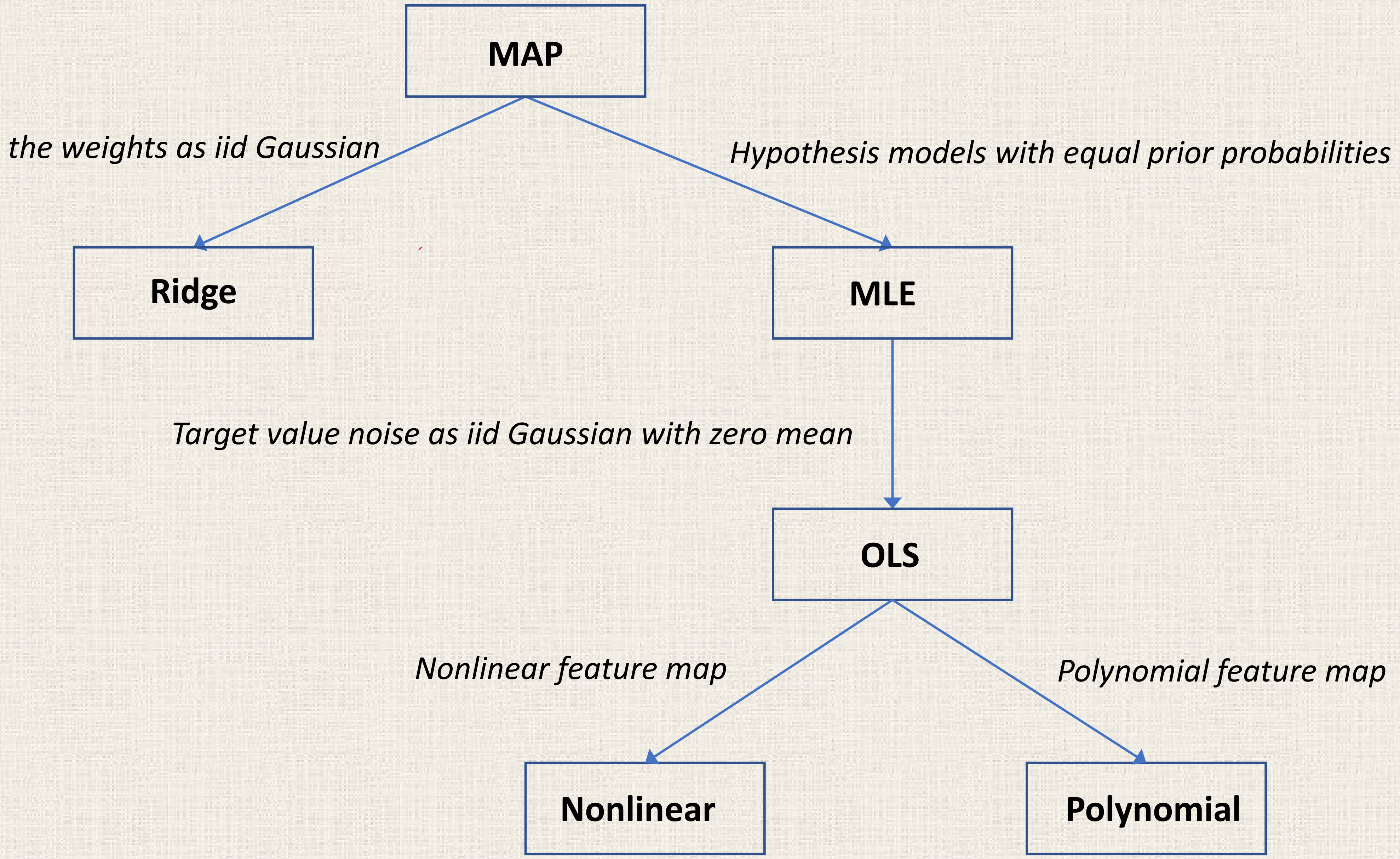
- Let $\lambda = \frac{\sigma^2}{\sigma_h^2}$, the above becomes:

$$\hat{\boldsymbol{w}}_{MAP} = \underset{\boldsymbol{w}}{argmin} \left\{ \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_{\mathbf{2}}{}^2 + \lambda \|\boldsymbol{w}\|_{\mathbf{2}}{}^2 \right\}$$

  This is just the ***Ridge Regression*** model!

- ***MAP is a probabilistic justification for adding the penalized term in Ridge Regression!***

- ***MAP model is robust to overfitting!***


MAP
the weights as iid Gaussian
Hypothesis models with equal prior probabilities
Ridge
MLE
Target value noise as iid Gaussian with zero mean
OLS
Nonlinear feature map
Polynomial feature map
Nonlinear
Polynomial