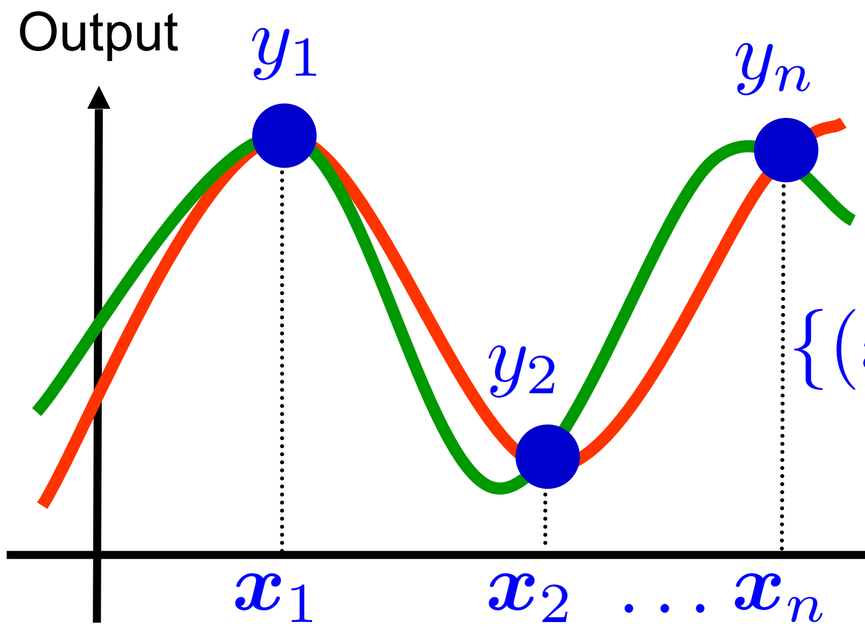
Least Squares

Masashi Sugiyama, Takashi Ishida
sugi@k.u-tokyo.ac.jp, ishi@k.u-tokyo.ac.jp
http://www.ms.k.u-tokyo.ac.jp

Schedule

1. 04/8 Introduction
2. 04/15 Regression 1
3. 04/22 Regression 2
- 04/30 Cancelled
4. 05/13 Classification 1
5. 05/20 Classification 2
6. 05/27 Deep learning 1
- 06/03 No lecture
7. 06/10 Deep learning 2
8. 06/17 Deep learning 3
9. 06/24 Semi-supervised learning
10. 07/01 Language models
11. 07/08 Representation learning 1
12. 07/15 Representation learning 2
13. 07/22 Advanced topics

Regression = Function Approximation



$f(\boldsymbol{x})$: function to be learned

$\hat{f}(\boldsymbol{x})$: learned function

$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$: training samples

$$y_i = f(\boldsymbol{x}_i)\ (+\text{noise})$$

Learn a function that is close to the underlying function by using training samples

Contents

Linear/non-linear model

- Model: candidate set to search for a learned function
  - A model is specified when we choose parameter $\boldsymbol{\theta}$

$$\{f_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\theta} = (\theta_1, \ldots, \theta_b)^\top\}$$

- Linear model: $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is linear w.r.t. $\boldsymbol{\theta}$

  (note: does not mean linear w.r.t. the input $\boldsymbol{x}$)
- Non-linear model: other models

Contents

Linear model

b functions

- Usual form: $$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$$ $\boldsymbol{x} \in \mathbb{R}^d$

  θ_j is j-th element of $\boldsymbol{\theta}$

- $\{\boldsymbol{\phi}_j(\boldsymbol{x})\}_{j=1}^{b}$: known basis functions that are linearly independent

- Examples of basis functions when $d = 1$:
  - Identity: $\phi(x) = x$ (model becomes linear in input)
  - Polynomial functions: $(1,\ x,\ x^2,\ \ \ldots,\ x^{b-1})$
  - Sinusoidal functions: $(1, \sin x, \cos x, \ldots, \sin kx, \cos kx)$

$b = 2k + 1$

Multi-dimensional linear model

- Additive model:

$$\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$$

T : transpose

$d \geq 2$

Ref: $\boldsymbol{x}_i$ is i-th sample

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{k=1}^{d} \sum_{j=1}^{b'} \theta_{k,j} \phi_j(x^{(k)})$$

- Number of parameters: $b = b'd$
  - Increases linearly w.r.t. input dimension d.
  - Note: Multiplicative model (in the next slide) grows exponentially w.r.t. d.
- Since it only considers the sum of one-dimensional basis functions, it cannot represent complex functions with interactions.

Multi-dimensional linear model

- Multiplicative model for multi-dimensional inputs:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j_1=1}^{b'} \cdots \sum_{j_d=1}^{b'} \theta_{j_1,\ldots,j_d} \phi_{j_1}(x^{(1)}) \cdots \phi_{j_d}(x^{(d)})$$

$$\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$$

- # of Params: $b = (b')^d$
  - Exponentially grows w.r.t. input dim d
  - $b = 10^{100}$ when $b' = 10, \quad d = 100$
  - Not practical from a computational/statistical perspective
- Can only be used when d is small (We will discuss about sparsity in the sparse learning lecture!)

Contents

Kernel model

- Linear model (that we discussed so far):
  - Basis functions $\{\phi_j(\boldsymbol{x})\}_{j=1}^{b}$ do not depend on the training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$
- Kernel model:
  - We use training input samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$ to design the basis functions

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

  - Ex: Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

$\|\boldsymbol{a}\|^2 = \boldsymbol{a}^\top \boldsymbol{a}$

$h(>0)$: bandwidth

Kernel model

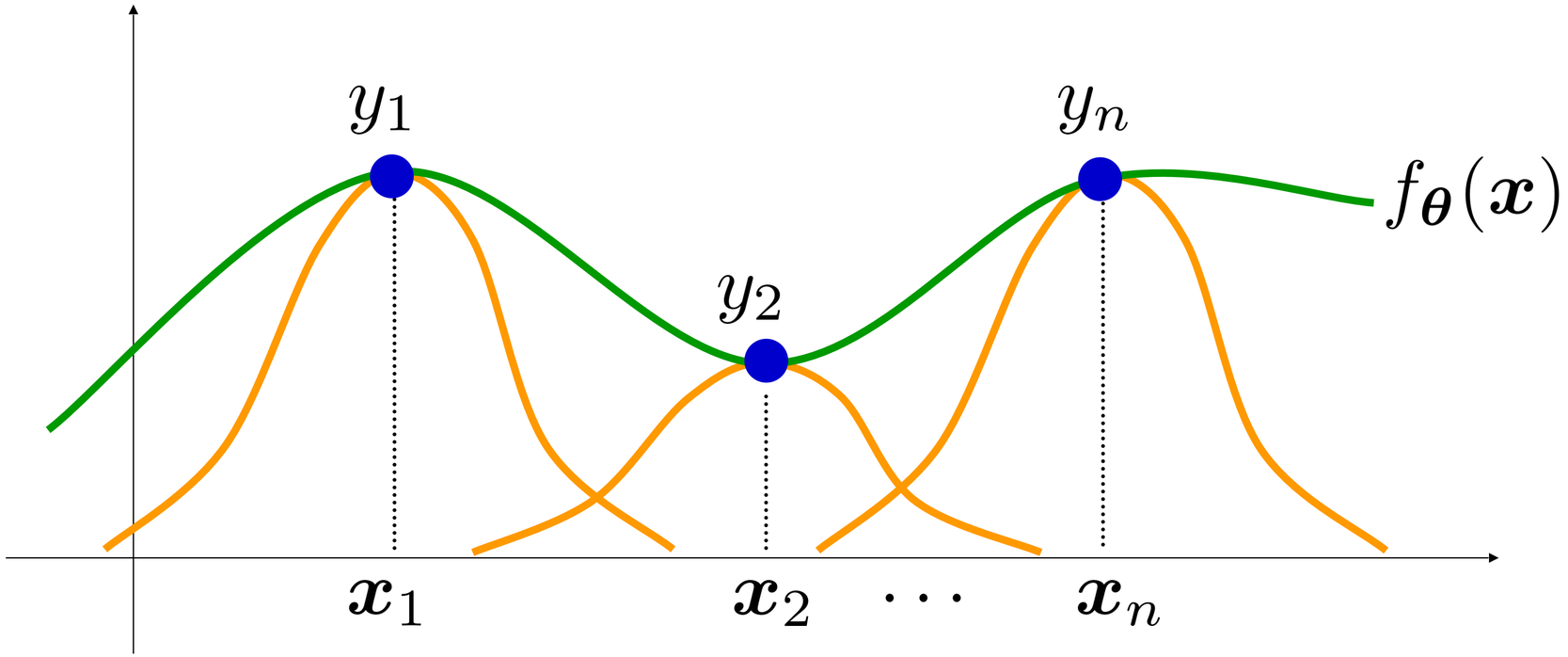
$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

- The # of params is n and does not depend on input dim d
- Linear in parameter:
  - kernel models is a type of linear model
- However, the # of params depend on the # of training samples, so have different statistical properties. Sometimes called nonparametric models due to this reason.
- In the scope of this course, you can treat kernel models as linear models.

Gaussian kernel model

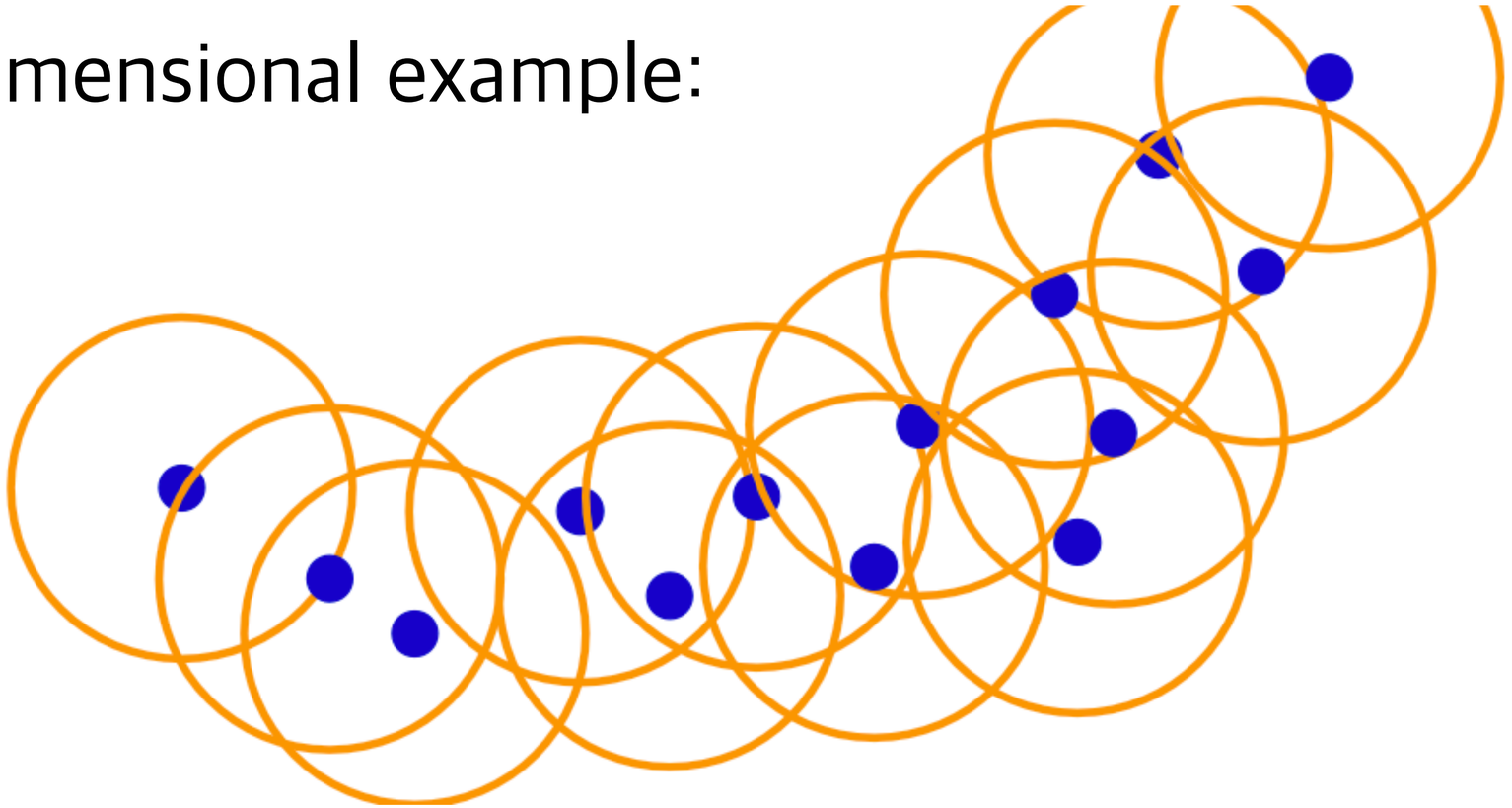
$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|^2}{2h^2}\right)$$

- Place Gaussian functions on each training input samples.



Gaussian kernel model

- When the n training samples are biased in their distribution across the input space, the Gaussian kernel model automatically ignores regions where there are no training input samples.
- A 2-dimensional example:



Contents

Non-linear model

- Models that are not linear w.r.t. the parameters
  - An example: a linear model with the basis functions including the parameter.

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \alpha_j \phi(\boldsymbol{x}; \boldsymbol{\beta}_j)$$

$$\boldsymbol{\theta} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_b^\top)^\top$$

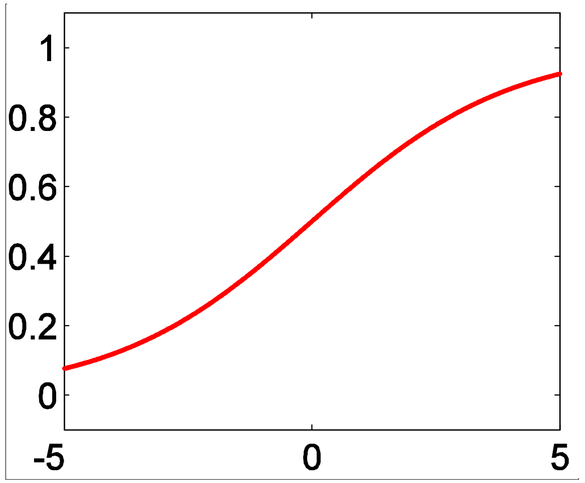
$\boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{\beta})$: non-linear w.r.t. $\boldsymbol{\beta}$

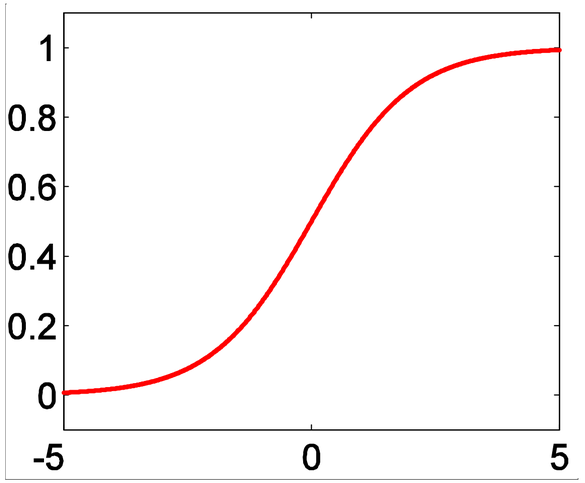
Ex 1: basis function that includes param

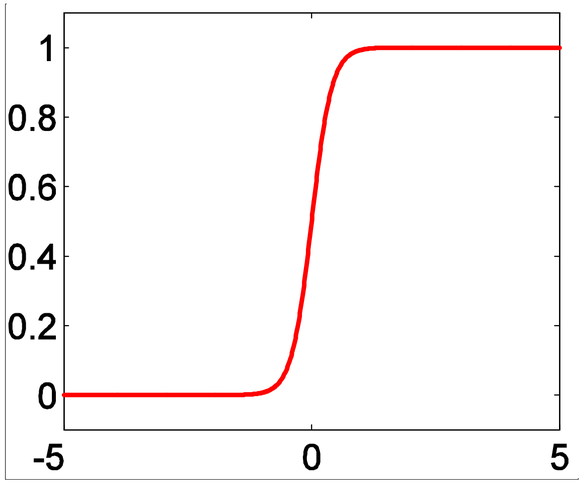
- Sigmoid function:

$$\phi(\boldsymbol{x};\boldsymbol{\beta}) = \frac{1}{1+\exp\left(-\boldsymbol{x}^\top \boldsymbol{w} - \gamma\right)} \qquad \boldsymbol{\beta} = (\boldsymbol{w}^\top, \gamma)^\top$$

For $d = 1$ and $\gamma = 0$:







w: small

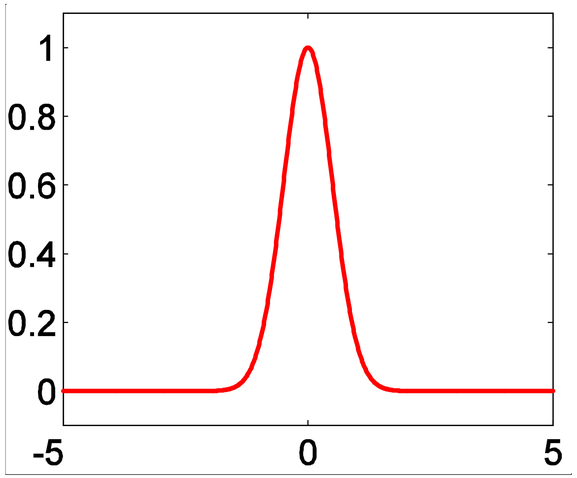
w: medium

w: large

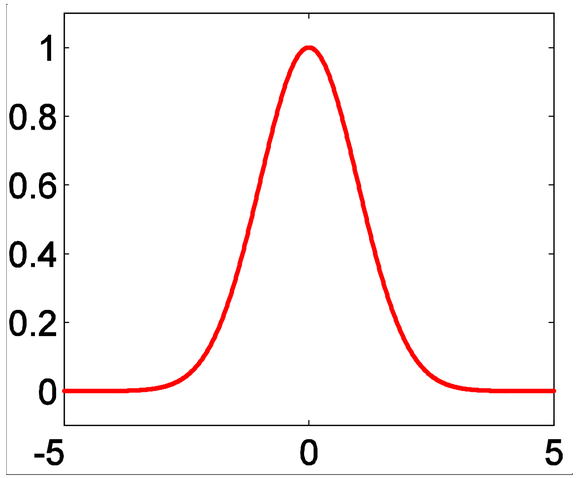
Ex 2: basis function that includes param

- Gaussian kernel: In contrast to linear Gaussian kernel models, we can learn the center and bandwidth of the Gaussian function

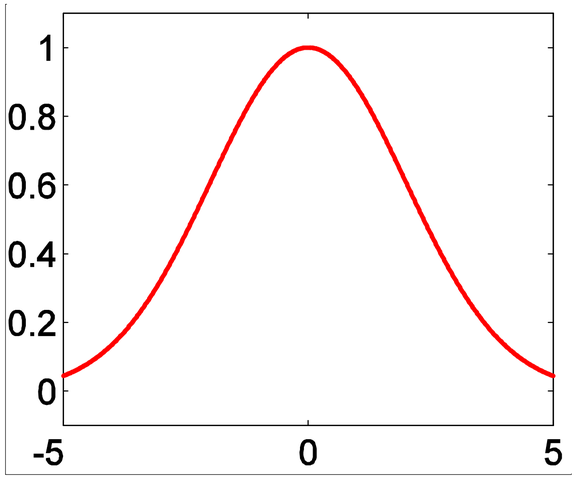
$$\phi(\boldsymbol{x};\boldsymbol{\beta}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|^2}{2h^2}\right) \qquad \boldsymbol{\beta} = (\boldsymbol{c}^\top, h)^\top$$



h: small



h: medium



h: large

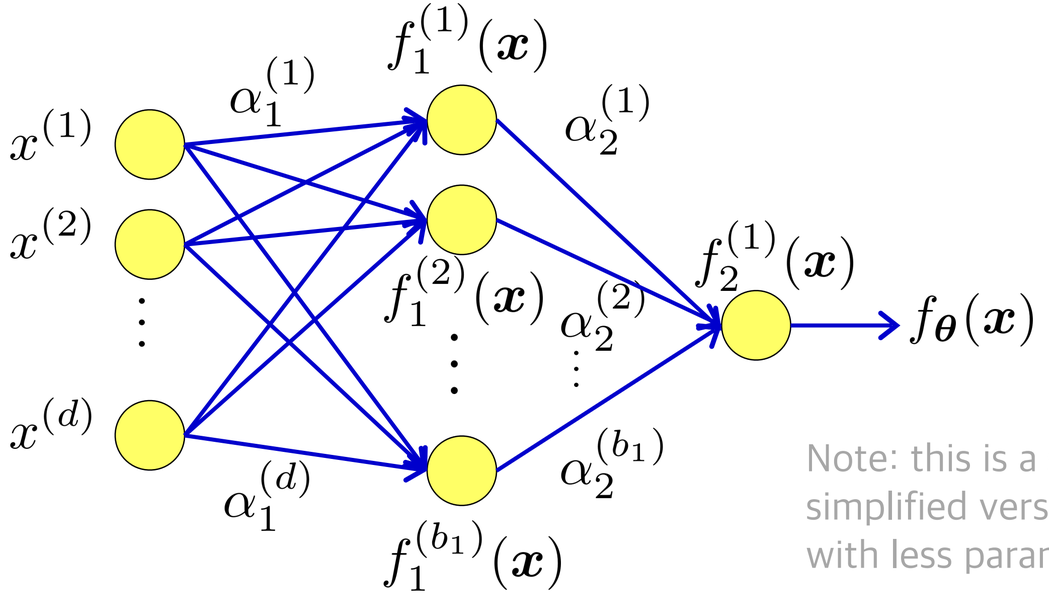
Non-linear model

- Model that is non-linear w.r.t. the parameters
  - Hierarchical model

$$f_k^{(i)}(\boldsymbol{x}) = \sum_{j=1}^{b_{k-1}} \alpha_k^{(j)} \phi(f_{k-1}^{(j)}(\boldsymbol{x})), \quad f_0^{(i)}(\boldsymbol{x}) = x^{(i)}$$

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_K^{(1)}(\boldsymbol{x})$$

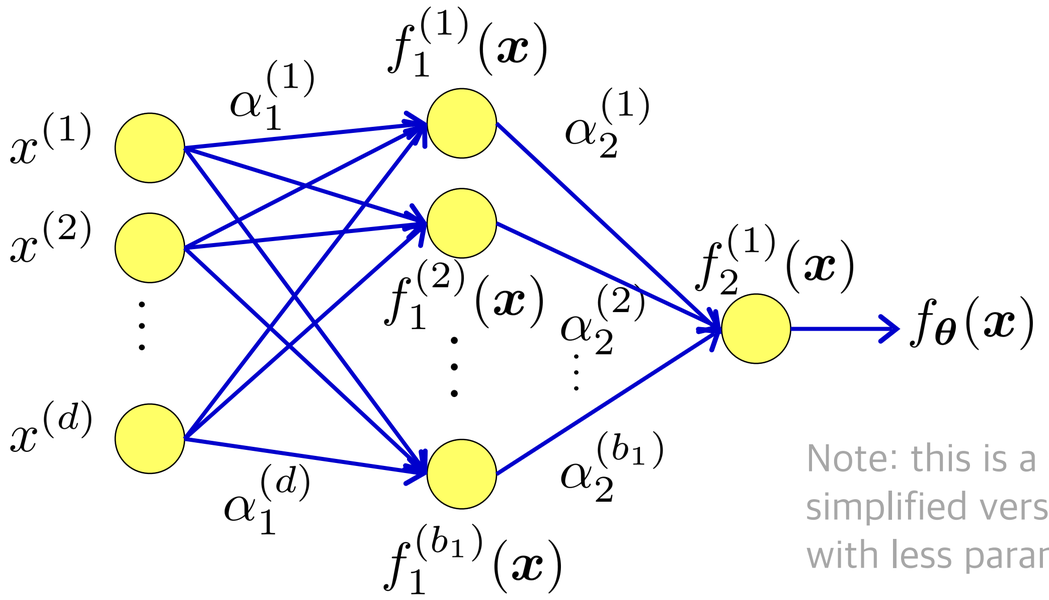
$$\boldsymbol{\theta} = (\boldsymbol{\alpha}_1^\top, \ldots, \boldsymbol{\alpha}_K^\top)^\top$$



Note: this is a slightly simplified version of an MLP with less parameters.

Some history on perceptrons

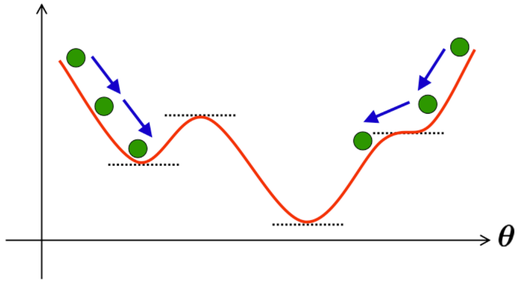
- The human brain consists of countless nerve cells connected in a network-like pattern.
- Sigmoid function behaves similarly to the nerve cells.
- Artificial neural networks are also called perceptrons.
- Mathematically, a 3-layer perceptron can approximate any function.



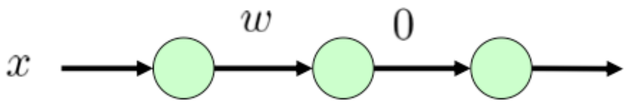
Note: this is a slightly simplified version of an MLP with less parameters.

Difficulty of training hierarchical models

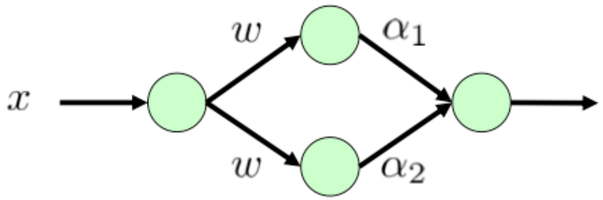
Historically, hierarchical models were **difficult to train**!



- Due to the existence of numerous local optima, finding the global optimum is difficult.
- Parameters and functions do not have a one-to-one correspondence.
  - When the weights of the second layer are zero, changing the weights of the first layer does not affect the function.



  - When the weights of the first layer are equal, if the sum of the weights of the second layer $\alpha_1 + \alpha_2$ is a constant, the function remains the same.



Modern deep learning largely mitigate these training difficulties, and we will discuss this in later lectures.

Summary of learning models

- Linear model: linear w.r.t. the parameters
  - Additive model: small number of params, low flexibility
  - Multiplicative model: High flexibility but too many params
  - kernel model: moderate flexibility and number of params
- Non-linear model: Non-linear w.r.t. to parameters
  - Hierarchical model: we will discuss this in the future
- We need to perform “model selection” since the decision will depend on the application.
- In this course: mainly use kernel models, occasionally hierarchical models such as neural network models.

Contents

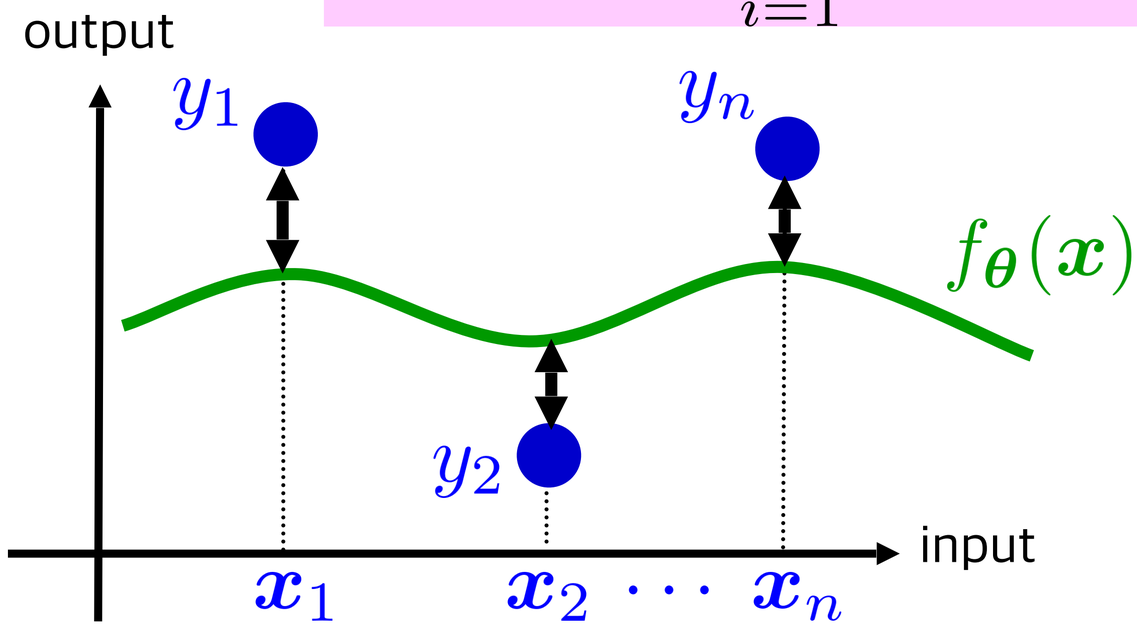
1. Learning model (Chap.2)
2. Least squares regression (Chap.3)
3. Regularized regression (Chap. 4)

Least squares regression

- Objective: minimize the squared error between the training outputs

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = \operatorname*{argmin}_{\boldsymbol{\theta}} J_{\mathrm{LS}}(\boldsymbol{\theta})$$

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\Big)^2$$



Deriving the solution

- Rewrite the objective for model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2$$

$$= \frac{1}{2} \|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|^2 = \frac{1}{2} (\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y})^\top (\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y})$$

$$\boldsymbol{y} = (y_1, \ldots, y_n)^\top$$

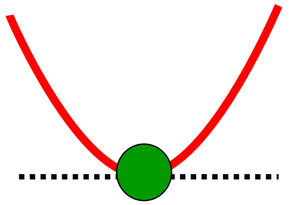
$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_b(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_n) & \cdots & \phi_b(\boldsymbol{x}_n) \end{pmatrix} : \text{Design matrix}$$

Deriving the solution (2)

- Partial derivative is zero:

$$\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}} = \left(\frac{\partial J_{\mathrm{LS}}}{\partial \theta_1}, \ldots, \frac{\partial J_{\mathrm{LS}}}{\partial \theta_b} \right)^\top = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta} - \boldsymbol{\Phi}^\top \boldsymbol{y} = \boldsymbol{0}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{\theta}^\top \boldsymbol{A} \boldsymbol{\theta} = 2 \boldsymbol{A} \boldsymbol{\theta} \qquad \frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{b}^\top \boldsymbol{\theta} = \boldsymbol{b}$$



- Equation satisfied by the solution:

$$\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{\theta} = \boldsymbol{\Phi}^\top \boldsymbol{y} \quad \Rightarrow \quad \widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$$

Note: assume $\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1}$ exists

- We analytically obtained the solution!

Why is $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ non-singular?

- Recall that $\{\phi_j(\boldsymbol{x})\}_{j=1}^b$ are known basis functions that are linearly independent. So the rank of $\boldsymbol{\Phi}$ is b.
- We first derive that $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is positive definite:
  - If $\boldsymbol{a} \equiv \boldsymbol{\Phi}\boldsymbol{c}$, then $\boldsymbol{c}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{c} = \boldsymbol{a}^\top\boldsymbol{a} = \sum_{i=1}^n a_i^2 \geq 0$.
  - Since $\boldsymbol{\Phi}$ is full column rank, $\boldsymbol{a} \neq \boldsymbol{0}$ for any $\boldsymbol{c} \neq \boldsymbol{0}$.
  - Therefore we have $\boldsymbol{c}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{c} > 0$.
- Since $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is positive definite, it is non-singular.
  - Proof by contradiction: If $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is singular, it means there exists a $\boldsymbol{c} \neq \boldsymbol{0}$ such that $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{c} = 0$. Then $\boldsymbol{c}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{c} = \boldsymbol{c}^\top\boldsymbol{0} = 0$.

Second-order condition

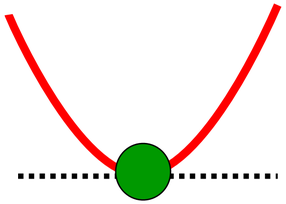
- To make sure the solution achieves a minimum instead of the maximum, we can check the second-order condition.
- We already derived:

$$\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}} = \boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{\Phi}^{\top}\boldsymbol{y}$$

- We can further derive the Hessian:

$$\nabla^2_{\boldsymbol{\theta}} J_{\mathrm{LS}} = \boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}$$

- From the previous slide, we know the Hessian is positive definite.

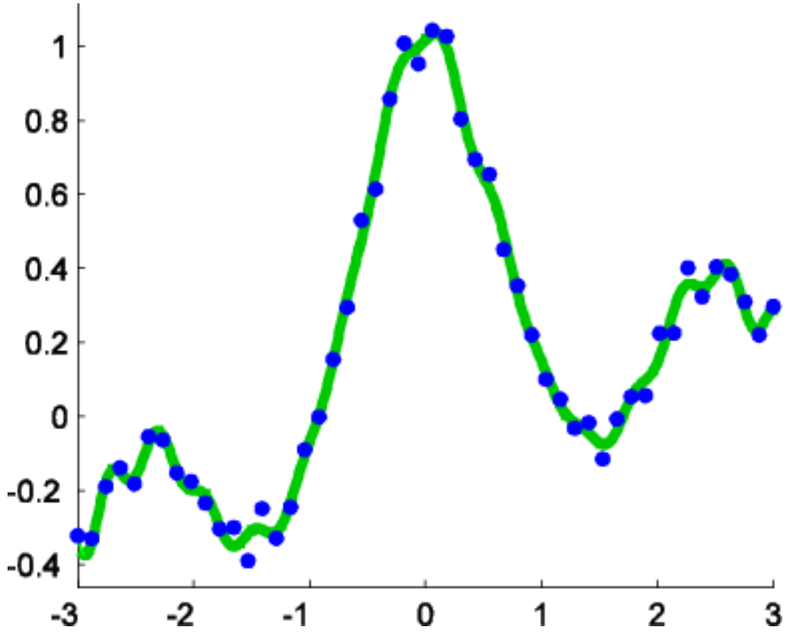


Example

- An example of least squares regression with a sinusoidal model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$

$$\phi_i(\boldsymbol{x}) = 1, \sin(x/2), \cos(x/2), \ldots, \sin(15x/2), \cos(15x/2)$$

true function: $f(x) = \sin(\pi x)/(\pi x) + 0.1x$



Example with Python


```
import numpy as np; import matplotlib; matplotlib.use('TkAgg'); import
matplotlib.pyplot as plt; np.random.seed(0)  # set seed for reproducibility

def generate_sample(xmin, xmax, sample_size):
    x = np.linspace(start=xmin, stop=xmax, num=sample_size)
    pix = np.pi * x
    target = np.sin(pix) / pix + 0.1 * x
    noise = 0.05 * np.random.normal(loc=0., scale=1., size=sample_size)
    return x, target + noise

def calc_design_matrix(x):
    sample_size = len(x)
    phi = np.empty(shape=(sample_size, 31))  # design matrix
    phi[:, 0] = 1.
    phi[:, 1::2] = np.sin(x[:, None] * np.arange(1, 16)[None] / 2)
    phi[:, 2::2] = np.cos(x[:, None] * np.arange(1, 16)[None] / 2)
    return phi

# create sample
sample_size = 50; xmin, xmax = -3, 3
x, y = generate_sample(xmin=xmin, xmax=xmax, sample_size=sample_size)

phi = calc_design_matrix(x) # calculate design matrix

theta = np.linalg.solve(phi.T.dot(phi), phi.T.dot(y[:, None])) # solve LS

# create data to visualize the prediction
X = np.linspace(start=xmin, stop=xmax, num=5000)
Phi = calc_design_matrix(X)
prediction = Phi.dot(theta)

# visualization
plt.clf(); plt.scatter(x, y, c='green', marker='o')
plt.plot(X, prediction); plt.show()
```

Exercise

- Prove that when the noise in the training output follows a normal distribution independently, the least squares regression is consistent with the maximum likelihood estimation of the Gaussian model.

$$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \quad y_i = f(\boldsymbol{x}_i) + \epsilon_i \quad \epsilon_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2)$$

- Probabilistic model: $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \dfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\dfrac{(y - f_{\boldsymbol{\theta}}(\boldsymbol{x}))^2}{2\sigma^2}\right)$
- Log-likelihood: $L(\boldsymbol{\theta}) = \sum_{i=1}^n \log p_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i)$
- Likelihood estimation: $\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^n \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2$$

Justification of least squares regression

- When the noise in the output follows a normal distribution independently:
  - Consistency: the least squares solution converges in probability to the optimal solution.
    - If there are an infinite number of samples, the optimal parameters can be obtained.
  - Asymptotic efficiency: has the smallest asymptotic variance among asymptotically normal estimators.
    - When there are a sufficiently large number of samples, the estimation results are stable.

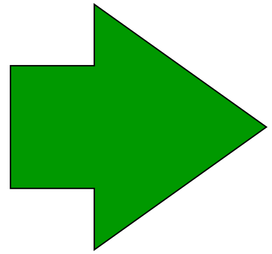
Least squares regression w/ kernel models

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

- Least squares objective:

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{K}\boldsymbol{\theta} - \boldsymbol{y}\|^2$$



$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = \boldsymbol{K}^{-1}\boldsymbol{y}$$

Kernel matrix:

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

Details of derivation

- Objective and solution:

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{K}\boldsymbol{\theta} - \boldsymbol{y}\|^2 \quad \Longrightarrow \quad \widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = \boldsymbol{K}^{-1}\boldsymbol{y}$$

- Since partial derivative is zero, we can derive:

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = (\boldsymbol{K}^\top \boldsymbol{K})^{-1}\boldsymbol{K}^\top \boldsymbol{y}$$

- A property of Kernel matrix: $\boldsymbol{K}^\top = \boldsymbol{K}$
- Hence,

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LS}} = (\boldsymbol{K}^2)^{-1}\boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}^{-1}\boldsymbol{y}$$

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix} \qquad K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

Summary of least squares regression

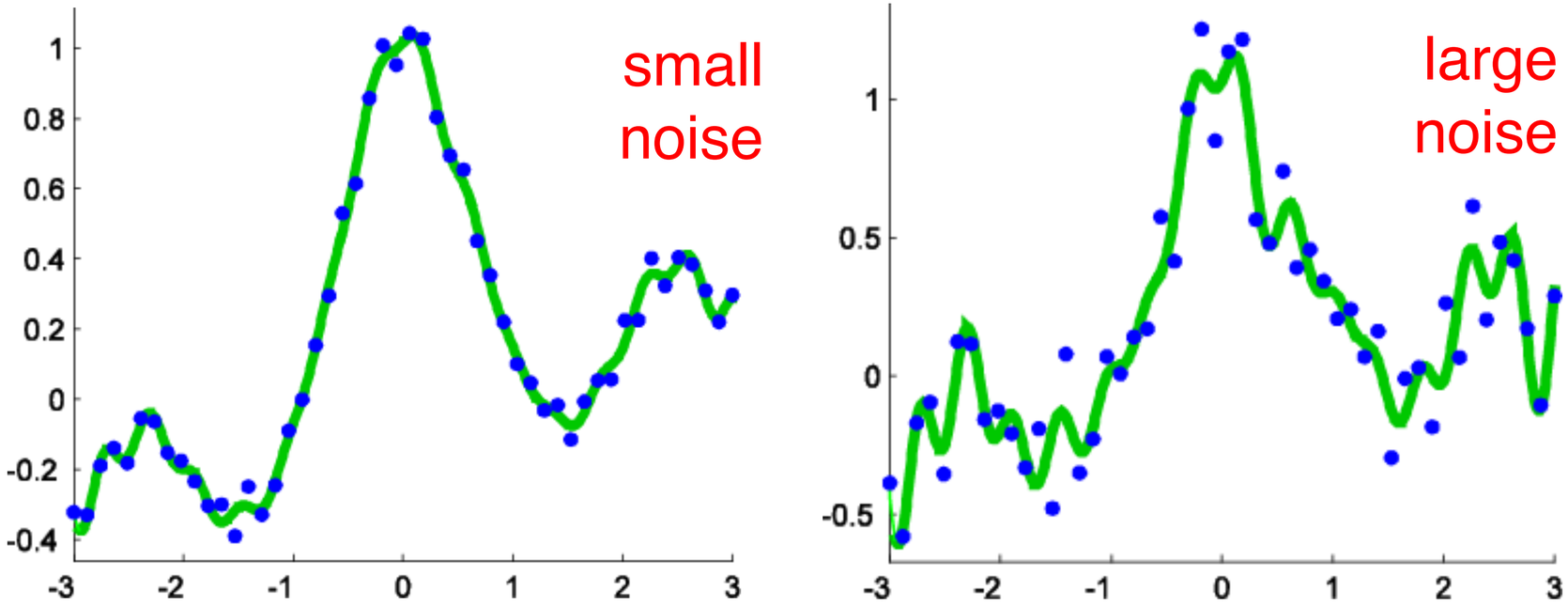
- **Main objective**: minimize the squared error between the training output and the predictions.
- **Regression**: basic technique for modeling the relationship between input and output variables.
- **When noise in the output is Gaussian**: the method can be interpreted as maximum likelihood estimation.
- **Analytical solutions**: for linear models and kernel models, we can find the optimal parameters analytically.

Contents

1. Learning model (Chap.2)
2. Least squares regression (Chap.3)
3. Regularized regression (Chap. 4)
   A) Constrained least squares regression
   B) Model selection

Issue of least squares regression

- Overfitting: occurs when a model adapts too much to the noise in the training samples.



$$\phi_i(\boldsymbol{x}) = 1, \sin x/2, \cos x/2, \ldots, \sin 15x/2, \cos 15x/2$$

➡ properly constrain the model to prevent this

Contents

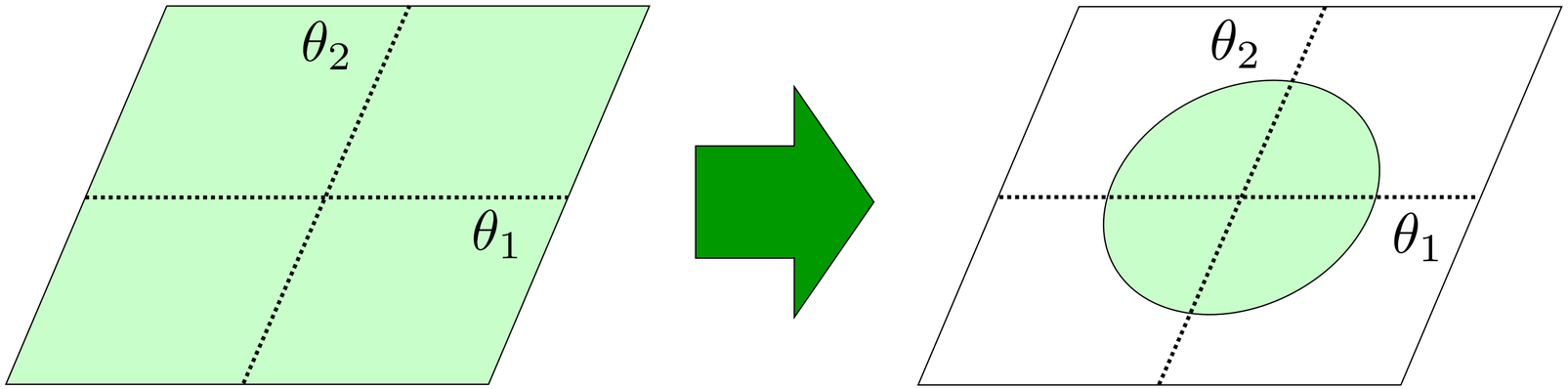
1. Learning model (Chap.2)
2. Least squares regression (Chap.3)
3. Regularized regression (Chap. 4)
   A) Constrained least squares regression
   B) Model selection

ℓ_2 –restricted least squares regression

- Avoid overfitting by restricting the model to be within a hyper-cube.
  (Because we want the absolute values of each elements of $\boldsymbol{\theta}$ to be small!)

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2 \quad \text{subject to } \|\boldsymbol{\theta}\|^2 \leq R$$

$R \geq 0$ $\quad f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$



Ordinary least squares regression

ℓ_2 -restricted least squares regression

Deriving the solution

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2 \quad \text{subject to } \|\boldsymbol{\theta}\|^2 \leq R$$

- Equivalent expression:

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{i=1}^{n} \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \right]$$

  - $\lambda (> 0)$: constant that is specified by R
- We can use λ instead of R in practice.

Interpretation

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \right]$$

goodness of fit for training output

penalty term for preventing param values from becoming too large (regularization)

- The model strikes a good balance between fitting the training output well and keeping the parameter values small.
- Also called ℓ_2-regularized regression

Exercise

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \right]$$

- Find the solution for linear model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$
- Info: $\frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2 = \frac{1}{2} \|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|^2$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_b(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_n) & \cdots & \phi_b(\boldsymbol{x}_n) \end{pmatrix} \qquad \boldsymbol{y} = (y_1, \ldots, y_n)^\top$$

Example

- (Gaussian) kernel model

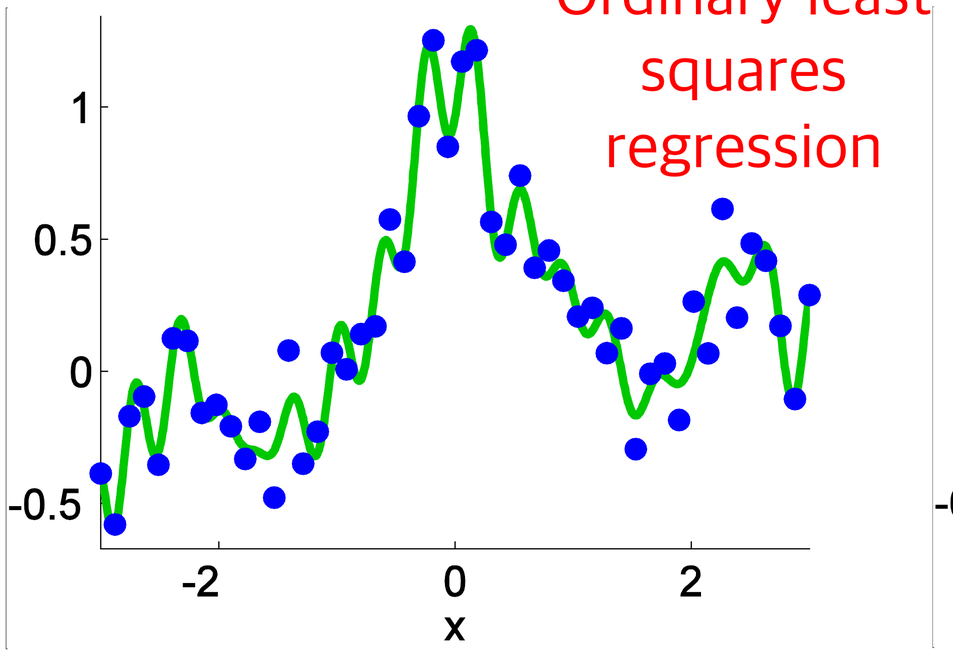
$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

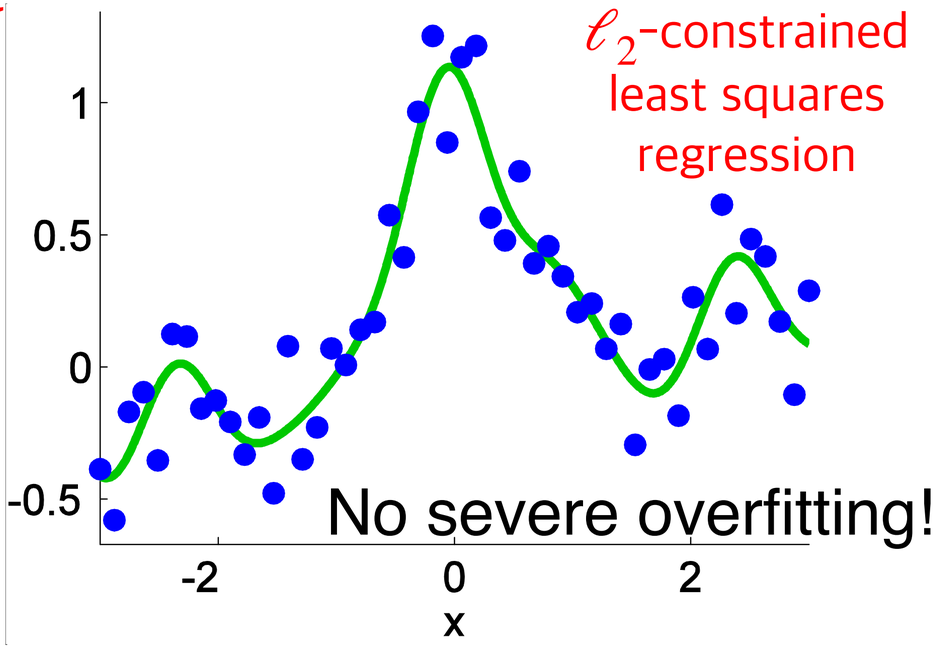
$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{K}^2 + \lambda \boldsymbol{I})^{-1} \boldsymbol{K}^\top \boldsymbol{y}$$

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

$$\boldsymbol{y} = (y_1, \ldots, y_n)^\top$$





Example with Python

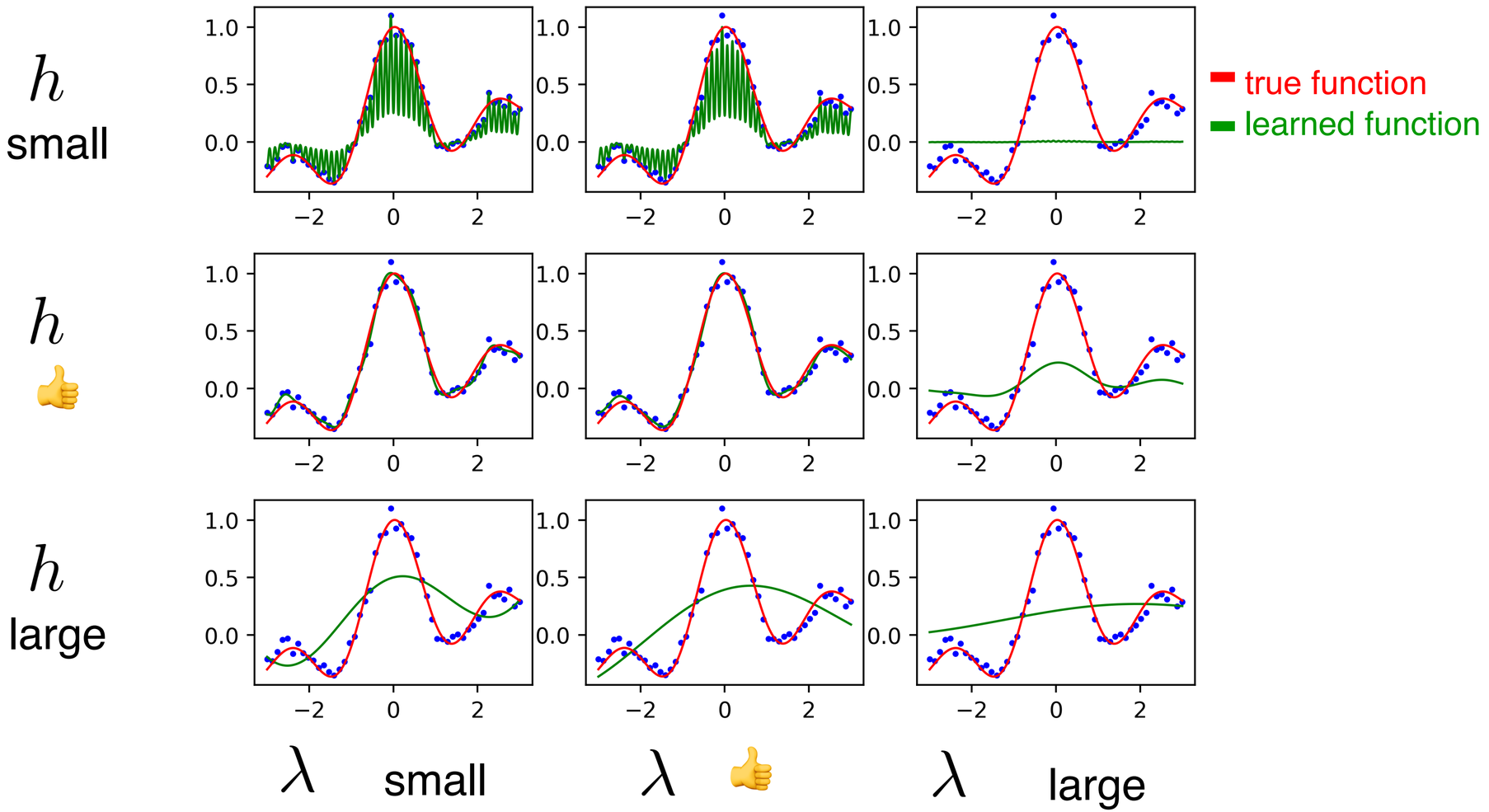
```
import numpy as np; import matplotlib; matplotlib.use('TkAgg')
import matplotlib.pyplot as plt; np.random.seed(0)  # set seed for reproducibility

def generate_sample(xmin, xmax, sample_size):
    x = np.linspace(start=xmin, stop=xmax, num=sample_size)
    pix = np.pi * x
    target = np.sin(pix) / pix + 0.1 * x
    noise = 0.05 * np.random.normal(loc=0., scale=1., size=sample_size)
    return x, target + noise

def calc_design_matrix(x, c, h):
    return np.exp(-(x[None] - c[:, None]) ** 2 / (2 * h ** 2))

# create sample
sample_size = 50
xmin, xmax = -3, 3
x, y = generate_sample(xmin=xmin, xmax=xmax, sample_size=sample_size)
# calculate design matrix
h = 0.1
k = calc_design_matrix(x, x, h)
# solve the least square problem
l = 0.3
theta = np.linalg.solve(
    k.T.dot(k) + l * np.identity(len(k)),
    k.T.dot(y[:, None]))
# create data to visualize the prediction
X = np.linspace(start=xmin, stop=xmax, num=5000)
K = calc_design_matrix(x, X, h)
prediction = K.dot(theta)
# visualization
plt.clf(); plt.scatter(x, y, c='green', marker='o')
plt.plot(X, prediction); plt.show()
```

Contents

1. Learning model (Chap.2)
2. Least squares regression (Chap.3)
3. Regularized regression (Chap. 4)
   A) Constrained least squares regression
   B) Model selection

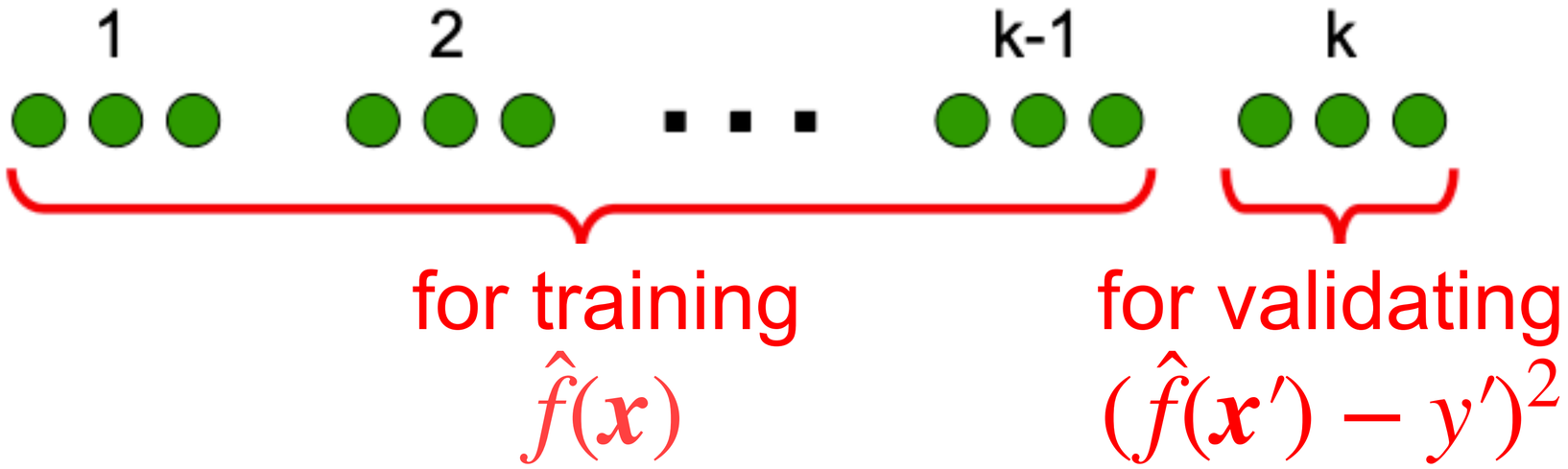
Model selection w/ regularized regression

- The results of regularized regression depends on how you choose the regularization param λ and Gaussian bandwidth h.



Cross validation

- Split **training** samples $Z = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into k groups: $\{Z_i\}_{i=1}^k$
- Use samples from groups excluding Z_i and learn θ (fix λ, h).
- Use the remaining Z_i to check the test error.
- Repeat this for all $i \in [k]$, and return the mean of the test errors.



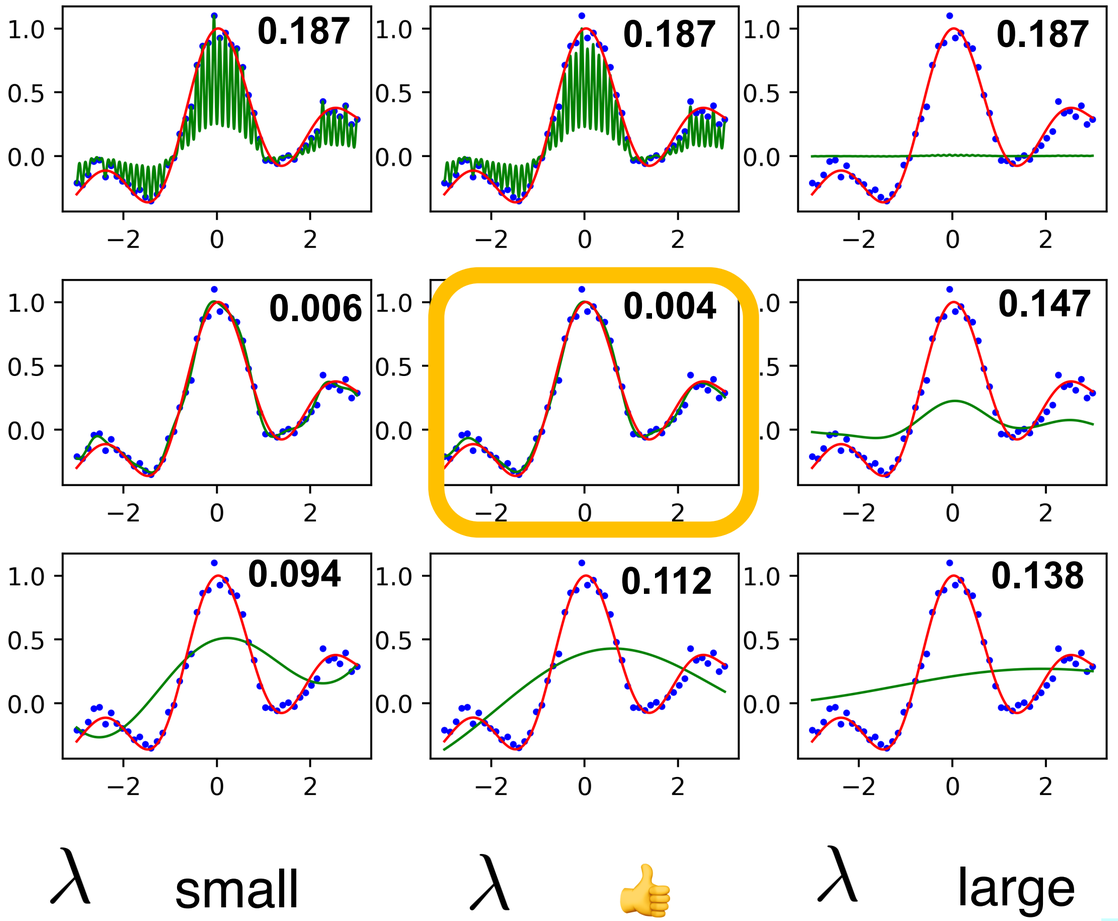
Example of CV

- Gaussian kernel model: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|^2}{2h^2}\right)$

h small

h 👍

h large



Homework for this week is implementing this.

Leave-one-out CV

- # of groups k = # of samples n

$$\frac{1}{n}\sum_{i=1}^{n}\left(\widehat{f}_i(\boldsymbol{x}_i)-y_i\right)^2$$

$\hat{f}_i(\boldsymbol{x})$: learned from data other than $(\boldsymbol{x}_i, y_i)$

  - Leave one sample out each time
- In general, regularized regression can be computationally expensive and may not be practical.
- However, for linear models, there are some regularized regression techniques that can be computed analytically.

Proving this is homework.

$$\frac{1}{n}\|\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}\boldsymbol{y}\|^2$$

$$\boldsymbol{H}=\boldsymbol{I}-\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^\top$$

$\tilde{\boldsymbol{H}}$: diagonals are same as $\boldsymbol{H}$, non-diagonals are zero

Summary of regularized regression

- Least squares regression can easily overfit to noise.
- Constrain the search range of parameters using the ℓ_2-norm.
- Solution can be analytically derived.
- Model selection is important.

Contents

1. Learning model (Chap.2)
2. Least squares regression (Chap.3)
3. Regularized regression (Chap. 4)

Summary

- Models for learning a function:
  - Linear models, kernel models, non-linear models
- Least squares regression:
  - Minimizes the squared error between training samples
  - The solution can be computed analytically
- Regularized regression:
  - Reduces overfitting in least squares regression
  - The solution can be computed analytically
  - Model selection is performed using cross-validation

Schedule

1. 04/8 Introduction
2. 04/15 Regression 1
3. 04/22 Regression 2
- 04/30 Cancelled
4. 05/13 Classification 1
5. 05/20 Classification 2
6. 05/27 Deep learning 1
- 06/03 No lecture
7. 06/10 Deep learning 2
8. 06/17 Deep learning 3
9. 06/24 Semi-supervised learning
10. 07/01 Language models
11. 07/08 Representation learning 1
12. 07/15 Representation learning 2
13. 07/22 Advanced topics

Homework 1

- Implement cross-validation for ℓ_2-regularized least squares regression with a Gaussian kernel model and determine the regularization parameter and Gaussian width.

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

Homework 2

- Show that the mean squared error for leave-one-out cross-validation of ℓ_2-regularized regression using a linear model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$ can be analytically calculated as follows:

$$\frac{1}{n} \| \widetilde{\boldsymbol{H}}^{-1} \boldsymbol{H} \boldsymbol{y} \|^2$$

$$\boldsymbol{H} = \boldsymbol{I} - \boldsymbol{\Phi} (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\top$$

$\tilde{\boldsymbol{H}}$: diagonals are same as $\boldsymbol{H}$, non-diagonals are zero

Homework 2 (Continued)

- Helpful information:
  - Express $\hat{\boldsymbol{\theta}}_i$ which is learned by removing $(\boldsymbol{x}_i, y_i)$ with:

$$\boldsymbol{\phi}_i = (\phi_1(\boldsymbol{x}_i), \phi_2(\boldsymbol{x}_i), \ldots, \phi_b(\boldsymbol{x}_i))^\top$$

$$\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n)^\top,$$

$$\boldsymbol{y},\ y_i,\ \boldsymbol{U} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I}$$

  - A special case of Sherman-Morrison-Woodbury:

$$(\boldsymbol{U} - \boldsymbol{u}\boldsymbol{u}^\top)^{-1} = \boldsymbol{U}^{-1} + \frac{\boldsymbol{U}^{-1}\boldsymbol{u}\boldsymbol{u}^\top\boldsymbol{U}^{-1}}{1 - \boldsymbol{u}^\top\boldsymbol{U}^{-1}\boldsymbol{u}}$$

Homework 2 (Continued)

- Helpful information:
  - Express prediction for $y_{i'}$ $\boldsymbol{\phi}_i^\top \hat{\boldsymbol{\theta}}_i$ (where the parameter was learned by removing $(\boldsymbol{x}_i, y_i)$) with the following.

$$\boldsymbol{\phi}_i = (\phi_1(\boldsymbol{x}_i), \phi_2(\boldsymbol{x}_i), \ldots, \phi_b(\boldsymbol{x}_i))^\top$$

$$\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n)^\top,$$

$$\boldsymbol{y},\ y_i,\ \boldsymbol{U} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I}$$

  - Expanding on the denominator that appears in the inverse matrix formula and repeating the process of common denominator