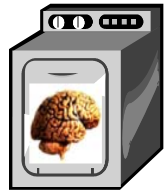
Pattern Information Processing: Linear Models and Least-Squares

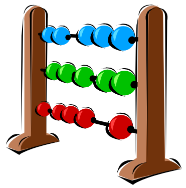
Masashi Sugiyama
(Department of Computer Science)

Contact: W8E-505
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

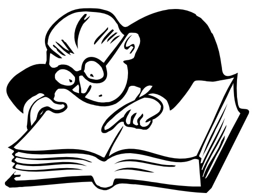
Focus of This Course

- There are 3 topics in learning research.
  - Understanding human brains
  - Developing learning machines
  - Mathematically clarifying mechanism of learning

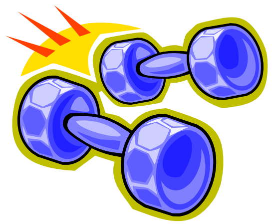




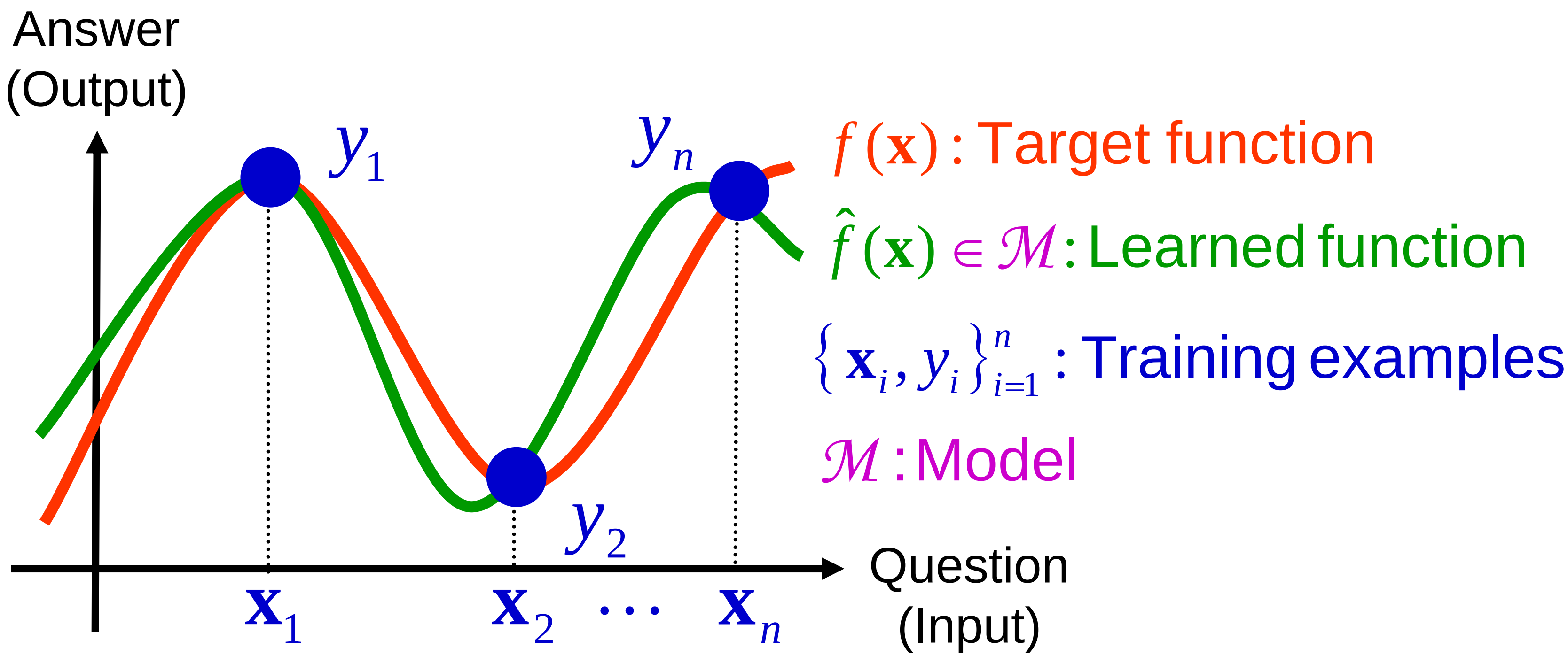
- There are 3 types of learning.
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning



- Topics of supervised learning:
  - Active learning
  - Model selection
  - Learning methods



Supervised Learning As Function Approximation



Using training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$,
find a function $\hat{f}(\mathbf{x})$ from a model $\mathcal{M}$
that well approximates the target function $f(\mathbf{x})$.

Formal Notation and Assumptions

- $\mathcal{D} \subset \mathbb{R}^d$: Input domain
- $f(\boldsymbol{x})$:Learning target function ($\mathcal{D} \to \mathbb{R}$)
- $\boldsymbol{x}_i \in \mathcal{D}$:Training input point
- $y_i = f(\boldsymbol{x}_i) + \epsilon_i$:Training output value
- ϵ_i :mean zero, independent and identically distributed ("i.i.d.")

$$\mathbb{E}_{\boldsymbol{\epsilon}}[\epsilon_i] = 0 \qquad \mathbb{E}_{\boldsymbol{\epsilon}}[\epsilon_i \epsilon_j] = \begin{cases} \sigma^2 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$:Training examples
- $\hat{f}(\boldsymbol{x})$:Learned function
- $\mathcal{M}$:Model

Generalization Error

- We want to obtain $\hat{f}(\boldsymbol{x})$ such that output values at unlearned test input points $\boldsymbol{t}$ can be accurately estimated.
- Suppose $\boldsymbol{t}$ follows a probability distribution with density $q(\boldsymbol{x})$.
- Expected test error (generalization error):

$$G = \int_{\mathcal{D}} \left(\hat{f}(\boldsymbol{t}) - f(\boldsymbol{t}) \right)^2 q(\boldsymbol{t}) d\boldsymbol{t}$$

- Goal: Obtain $\hat{f}(\boldsymbol{x})$ such that G is minimized.

Formal Description of Problems

$$G = \int_{\mathcal{D}} \left(\hat{f}(\boldsymbol{t}) - f(\boldsymbol{t}) \right)^2 q(\boldsymbol{t}) d\boldsymbol{t}$$

- Active learning: $\min_{\{\boldsymbol{x}_i\}_{i=1}^{n}} G$

- Model selection: $\min_{\mathcal{M}} G$

- Learning methods: $\min_{\hat{f} \in \mathcal{M}} G$

Today's Plan

- Models
  - Linear models
  - Kernel models
- Learning methods
  - Least-squares learning

Linear/Non-Linear Models

- Model is a set of functions from which learning result functions are searched.
- We use a family of functions $\hat{f}(\boldsymbol{x})$ parameterized by

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_b)^\top$$

- Linear model: $\hat{f}(\boldsymbol{x})$ is linear with respect to $\boldsymbol{\alpha}$

  (Note: not necessarily linear with respect to $\boldsymbol{x}$)
- Non-linear model: Otherwise

Linear Models

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

- $\{\varphi_i(\boldsymbol{x})\}_{i=1}^{b}$:Linearly independent functions
- For example, when $d = 1$
  - Polynomial

$$1, x, x^2, \ldots, x^{b-1}$$

  - Trigonometric polynomial

$$1, \sin x, \cos x, \ldots, \sin kx, \cos kx$$

$$b = 2k + 1$$

Multi-Dimensional Linear Models

- For multidimensional input $(d > 1)$, a product model could be used.

$$\hat{f}(\boldsymbol{x}) = \sum_{i_1=1}^{c} \sum_{i_2=1}^{c} \cdots \sum_{i_d=1}^{c}$$

$$\alpha_{i_1,i_2,\ldots,i_d} \varphi_{i_1}(x^{(1)}) \varphi_{i_2}(x^{(2)}) \cdots \varphi_{i_d}(x^{(d)})$$

$$\boldsymbol{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})^\top$$

- The number of parameters is $b = c^d$, which increases exponentially w.r.t. d .
- Infeasible for large d !

Additive Models

- For large d, we have to reduce the number of parameters.
- Additive model:

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{d} \sum_{i=1}^{c} \alpha_{i,j} \varphi_i(x^{(j)})$$

- The number of parameters is only $b = cd$.
- However, additive model is too simple so its representation capability may not be rich enough in some application.

Kernel Models

- Linear model:

  $\{\varphi_i(\boldsymbol{x})\}_{i=1}^{b}$ do not depend on $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$

- Kernel model: $$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

- $K(\boldsymbol{x}, \boldsymbol{x}')$:Kernel function
  - Suppose kernel is symmetric:

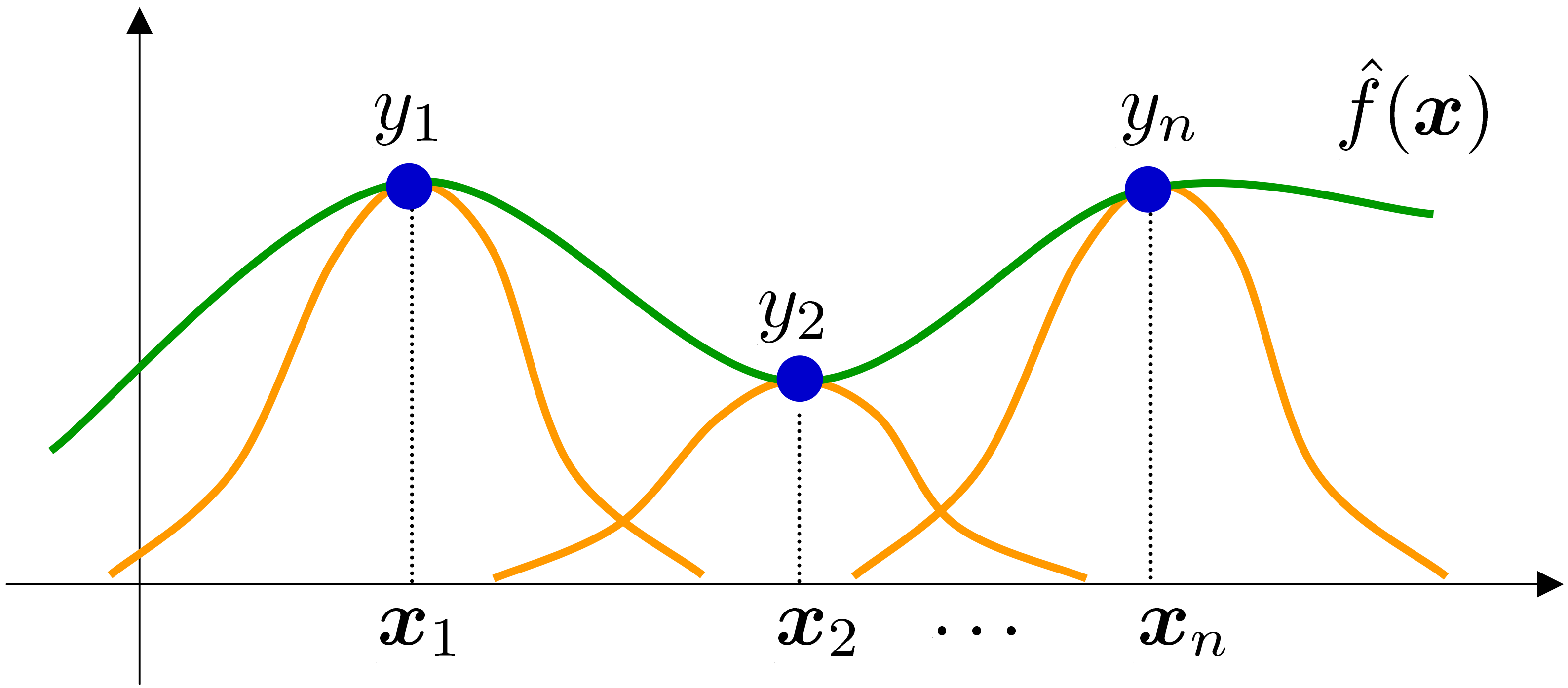
    $$K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})$$

  - e.g., Gaussian kernel

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

Kernel Models (cont.)

- Put kernel functions at training input points.



Kernel Models (cont.)

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

- The number of parameters is n, which is independent of the input dimensionality d.
- Although kernel model is linear w.r.t. $\boldsymbol{\alpha}$, the number of parameters grows as the number of training samples increases.
- Mathematical treatment could be different from ordinary linear models (called "non-parametric models" in statistics).

Summary of Linear Models

- Linear model (product):
  High flexibility, high complexity
- Linear model (additive):
  Low flexibility, low complexity
- Kernel model:
  Moderate flexibility, moderate complexity

- Good model depends on applications.
- Later in model selection, we discuss how to choose appropriate models.

Learning Methods

- Linear learning methods:

  Parameter vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_b)^\top$

  is estimated linearly w.r.t.

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top$$

- Non-linear learning methods: Otherwise

Linear Learning for Linear Models / Kernel Models

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

- In linear learning methods, a learned parameter vector is given by

$$\hat{\boldsymbol{\alpha}} = \boldsymbol{L}\boldsymbol{y}$$

$\boldsymbol{L}$:Learning matrix

Least-Squares Learning

- Learn $\boldsymbol{\alpha}$ such that the squared error at training input points is minimized:

$$\hat{\boldsymbol{\alpha}}_{LS} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b} J_{LS}(\boldsymbol{\alpha})$$

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\left(\hat{f}(\boldsymbol{x}_i) - y_i\right)^2$$

$$= \|\boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y}\|^2$$

$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$:Design matrix $(n \times b)$

- In the following, we assume $\mathrm{rank}\,(\boldsymbol{X}) = b$

How to Obtain Solutions

- Extreme-value condition:

$$\nabla J_{LS}(\hat{\boldsymbol{\alpha}}_{LS}) = 2\boldsymbol{X}^\top(\boldsymbol{X}\hat{\boldsymbol{\alpha}}_{LS} - \boldsymbol{y}) = 0$$

$$\Longrightarrow \quad \hat{\boldsymbol{\alpha}}_{LS} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

- Therefore, LS is linear learning.

$$\hat{\boldsymbol{\alpha}}_{LS} = \boldsymbol{L}_{LS}\boldsymbol{y}$$

$$\boldsymbol{L}_{LS} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top$$

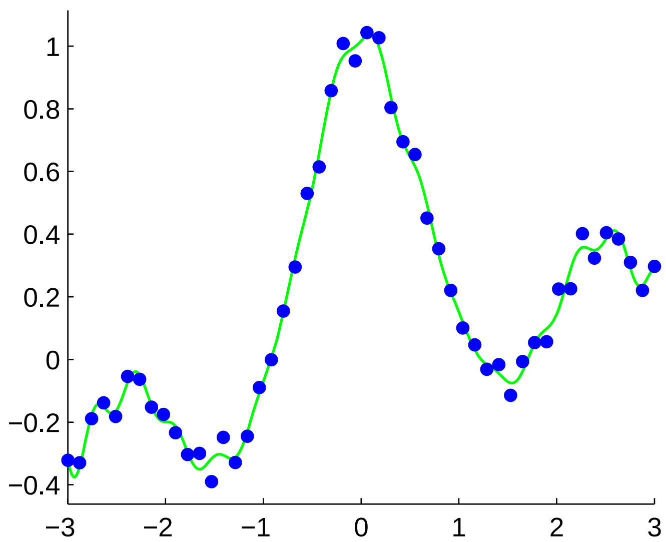
If you are not familiar with vector-derivatives, see e.g, "Matrix Cookbook" (http://matrixcookbook.com)

Example of LS

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

- Trigonometric polynomial model

$$1, \sin x, \cos x, \ldots, \sin 15x, \cos 15x \quad (b = 31)$$



Homework

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

1. Prove that the LS solution in kernel models is given by

$$\hat{\boldsymbol{\alpha}}_{LS} = \boldsymbol{L}_{LS} \boldsymbol{y}$$

$$\boldsymbol{L}_{LS} = \boldsymbol{K}^{-1}$$

$$\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

(Kernel matrix)

Homework (cont.)

2. For your own toy 1-dimensional data, perform simulations using
   - Gaussian kernel models
   - Least-squares learning

   and analyze the results when, e.g.,
   - Target functions
   - Number of samples
   - Noise level
   - Width of Gaussian kernel

   are changed.

- Deadline: May 11th

Coming Classes…

- April 27th: No class
- May 11th: Guest lecture by Dr. Hirotaka Hachiya
  - Machine learning and robotics
- May 18th: Guest lecture by Dr. Makoto Yamada
  - Machine learning and speech processing
- May 25th: Regular lecture