Pattern Information Processing: Model Selection by Cross-Validation

Masashi Sugiyama

(Department of Computer Science)

Contact: W8E-505

sugi@cs.titech.ac.jp

http://sugiyama-www.cs.titech.ac.jp/~sugi/

Model Parameters

- In the process of parameter learning, we fixed model parameters.
- For example, quadratically constrained least-squares with Gaussian kernel models:
  - Gaussian width: $c\ (>0)$
  - Regularization parameter: $\lambda\ (\geq 0)$

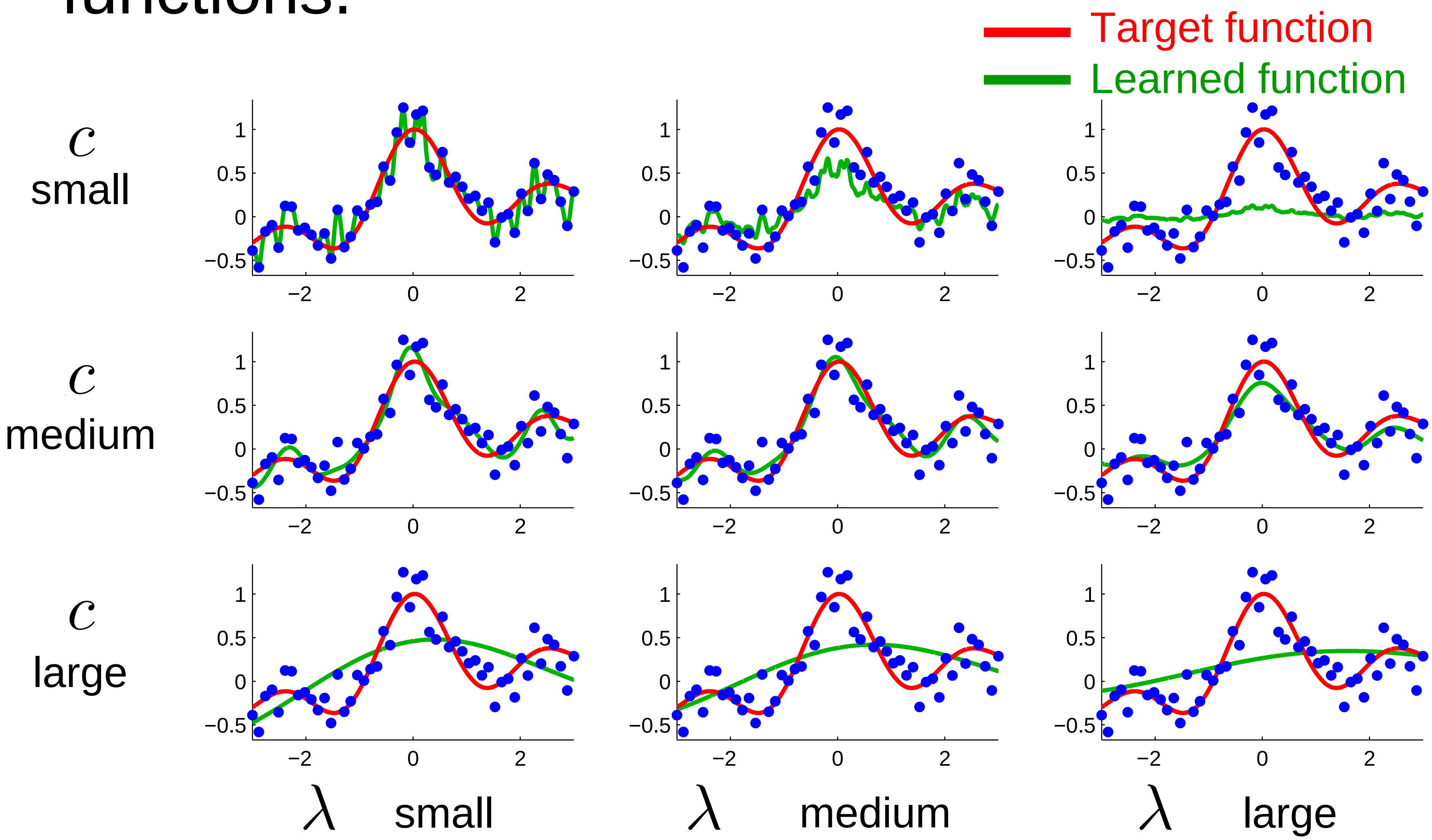
$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

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

$$J_{QCLS}(\boldsymbol{\alpha}) = J_{LS}(\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|^2$$

Different Model Parameters

- Model parameters strongly affect learned functions.



Determining Model Parameters

- We want to determine the model parameters so that the generalization error (expected test error) is minimized.

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

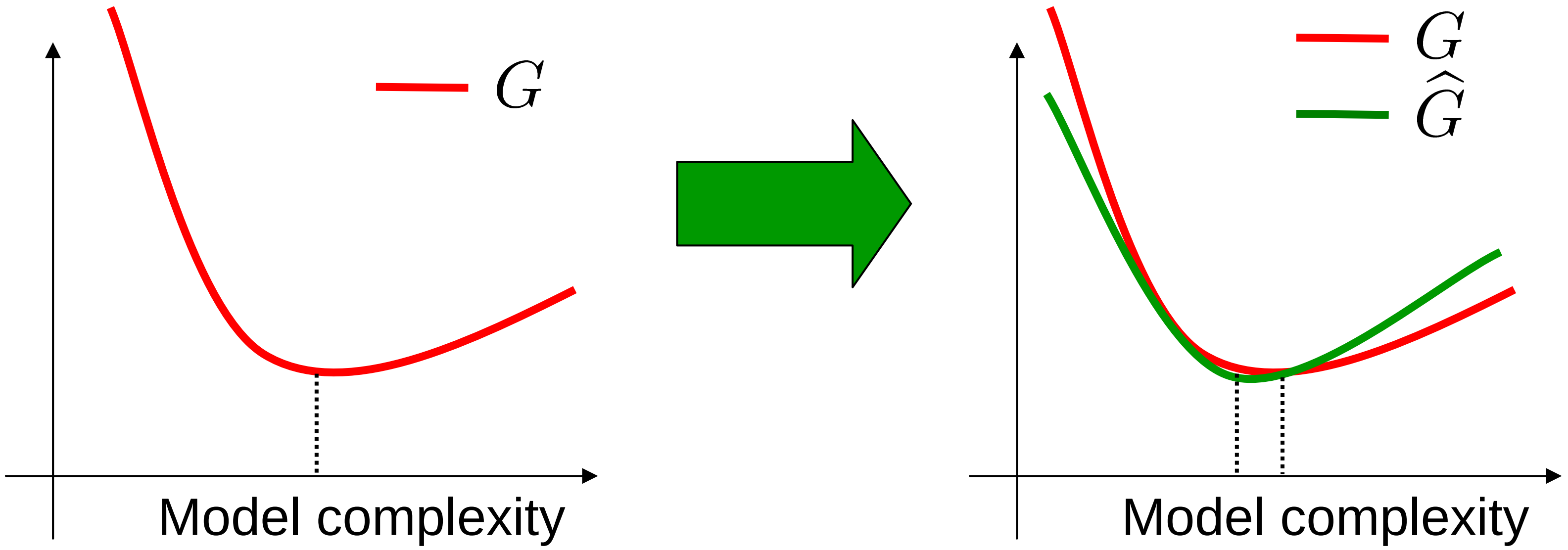
$$\boldsymbol{t} \sim q(\boldsymbol{x})$$

- However, $f(\boldsymbol{x})$ is unknown so the generalization error is not accessible.
- $q(\boldsymbol{x})$ may also be unknown.

Generalization Error Estimation

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

- Instead, we use a generalization error estimate.



Model Selection

- Prepare a set of model candidates.

$$\{\mathcal{M}_i \mid \mathcal{M}_i = (c_i, \lambda_i)\}$$

- Estimate generalization error for each model.

$$\widehat{G}(\mathcal{M}_i)$$

- Choose the one with the minimum estimated generalization error.

$$\widehat{\mathcal{M}} = \underset{\mathcal{M} \in \{\mathcal{M}_i\}_i}{\operatorname{argmin}} \ \widehat{G}(\mathcal{M})$$

Assumptions

- Training input points: $\boldsymbol{x}_i \overset{i.i.d.}{\sim} q(\boldsymbol{x})$
- Training output values: $y_i = f(\boldsymbol{x}_i) + \epsilon_i$
- Noise ϵ_i : i.i.d., mean 0, variance σ^2

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

Extra-Sample Method

- Suppose we have an extra example $(\boldsymbol{x}', y')$ in addition to $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.

$$\boldsymbol{x}' \sim q(\boldsymbol{x}) \qquad y' = f(\boldsymbol{x}') + \epsilon' \qquad \mathbb{E}_{\boldsymbol{\epsilon}}[\epsilon'] = 0$$

$$\mathbb{E}_{\boldsymbol{\epsilon}}[\epsilon'^2] = \sigma^2$$

$$\mathbb{E}_{\boldsymbol{\epsilon}}[\epsilon' \epsilon_i] = 0, \quad \forall i$$

- Test the prediction performance of the learned function using the extra example.

$$\widehat{G}_{extra} = \left(\hat{f}(\boldsymbol{x}') - y'\right)^2$$

$$\hat{f} \longleftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$$

Extra-Sample Method (cont.)

$\widehat{G}_{extra}$ is unbiased w.r.t. $\boldsymbol{x}'$ and ϵ' (up to σ^2)

$$\mathbb{E}_{\boldsymbol{x}'}\mathbb{E}_{\epsilon'}[\widehat{G}_{extra}] = G + \sigma^2$$

- Proof:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}'}\mathbb{E}_{\epsilon'}\left(\hat{f}(\boldsymbol{x}') - f(\boldsymbol{x}') - \epsilon'\right)^2 \\
&= \mathbb{E}_{\boldsymbol{x}'}\mathbb{E}_{\epsilon'}\left[(\hat{f}(\boldsymbol{x}') - f(\boldsymbol{x}'))^2 - 2\epsilon'(\hat{f}(\boldsymbol{x}') - f(\boldsymbol{x}')) + \epsilon'^2\right] \\
&= G + \sigma^2
\end{aligned}$$

- $\widehat{G}_{extra}$ may be used for model selection.
- However, in practice, such an extra example is not available (or if we have, it should be included in the original training set!).

Holdout Method

- Idea: Use one of the training samples as an extra sample

1. Divide training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ into $\{(\boldsymbol{x}_i, y_i)\}_{i \neq j}$ and $(\boldsymbol{x}_j, y_j)$.
2. Train a learning machine using $\{(\boldsymbol{x}_i, y_i)\}_{i \neq j}$

$$\hat{f}_j(\boldsymbol{x}) \longleftarrow \{(\boldsymbol{x}_i, y_i)\}_{i \neq j}$$

3. Test its prediction performance using the holdout sample $(\boldsymbol{x}_j, y_j)$:

$$\widehat{G}_j = \left(\hat{f}_j(\boldsymbol{x}_j) - y_j\right)^2$$

Almost Unbiasedness of Holdout

- Holdout method is almost unbiased w.r.t. $\boldsymbol{x}_j, \epsilon_j$:

$$\mathbb{E}_{\boldsymbol{x}_j}\mathbb{E}_{\epsilon_j}[\widehat{G}_j] = G_j + \sigma^2$$
$$\approx G + \sigma^2$$

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

$$\widehat{f}_j(\boldsymbol{x}) \approx \widehat{f}(\boldsymbol{x}) \text{ if } n \text{ is large}$$

- However, $\widehat{G}_j$ is heavily affected by the choice of the holdout sample $(\boldsymbol{x}_j, y_j)$.

Leave-One-Out Cross-Validation

- Repeat the holdout procedure for all combinations and output the average.

$$\widehat{G}_{LOOCV} = \frac{1}{n}\sum_{j=1}^{n}\widehat{G}_j$$

$$\widehat{G}_j = \left(\hat{f}_j(\boldsymbol{x}_j) - y_j\right)^2$$

- LOOCV is almost unbiased w.r.t. $\{\boldsymbol{x}_i, \epsilon_i\}_{i=1}^{n}$

$$\mathbb{E}_{\{\boldsymbol{x}_i\}_{i=1}^{n}}\mathbb{E}_{\{\epsilon_i\}_{i=1}^{n}}[\widehat{G}_{LOOCV}] \approx \mathbb{E}_{\{\boldsymbol{x}_i\}_{i=1}^{n}}\mathbb{E}_{\{\epsilon_i\}_{i=1}^{n}}[G] + \sigma^2$$

k-Fold Cross-Validation

- Randomly split training set into k disjoint subsets $\{\mathcal{T}_j\}_{j=1}^{k}$.

$$\widehat{G}_{kCV} = \frac{1}{k}\sum_{j=1}^{k}\widehat{G}_{\mathcal{T}_j}$$

$$\widehat{G}_{\mathcal{T}_j} = \frac{1}{|\mathcal{T}_j|}\sum_{i\in\mathcal{T}_j}\left(\hat{f}_{\mathcal{T}_j}(\boldsymbol{x}_i) - y_i\right)^2$$

$$\hat{f}_{\mathcal{T}_j}(\boldsymbol{x}) \longleftarrow \{(\boldsymbol{x}_i, y_i) \mid i \notin \mathcal{T}_j\}$$

- k-fold is easier to compute and more stable.

Advantages of CV

- Wide applicability: Almost unbiasedness of LOOCV holds for (virtually) any learning methods
- Practical usefulness: CV is shown to work very well in many practical applications

Disadvantages of CV

- Computationally expensive
  It requires repeating training of models with different subsets of training samples
- Number of folds
  It is often recommended to use $k = 5, 10$. However, how to optimally choose k is still open.

Closed Form of LOOCV

- Linear model

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

- Quadratically constrained least-squares

$$J_{QCLS}(\boldsymbol{\alpha}) = J_{LS}(\boldsymbol{\alpha}) + \lambda \|\boldsymbol{\alpha}\|^2$$

$$\widehat{G}_{LOOCV} = \frac{1}{n} \|\widetilde{\boldsymbol{H}}^{-1} \boldsymbol{H} \boldsymbol{y}\|^2$$

$$\boldsymbol{H} = \boldsymbol{I} - \boldsymbol{X} \boldsymbol{L}_{QCLS} \qquad \boldsymbol{L}_{QCLS} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top$$

$\widetilde{\boldsymbol{H}}$:same diagonal as $\boldsymbol{H}$ but zero for off-diagonal

Homework (cont.)

1. (Try to) prove the closed-form expression of leave-one-out cross-validation score for quadratically constraint least-squares.

$$\widehat{G}_{LOOCV} = \frac{1}{n}\|\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}\boldsymbol{y}\|^2$$

Hint: Express $\hat{\boldsymbol{\alpha}}_j$ in terms of $\hat{\boldsymbol{\alpha}}$

- $\hat{\boldsymbol{\alpha}}_j$:Learned parameter without the j-th sample
- $\hat{\boldsymbol{\alpha}}$:Learned parameter with all samples.

Key fact:

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

Homework (cont.)

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

   and optimize
   - Width of Gaussian kernel
   - Regularization parameter

   based on cross-validation. Analyze the results when changing
   - Target function
   - Number of samples
   - Noise level

Suggestions

- Please look for software which can solve
  - Linearly constrained quadratic programming

$$\min_{\boldsymbol{\beta}} \left[\frac{1}{2} \langle \boldsymbol{Q}\boldsymbol{\beta}, \boldsymbol{\beta} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \right]$$

$$\text{subject to } \boldsymbol{V}\boldsymbol{\beta} \leq \boldsymbol{v} \text{ and } \boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

  - Linearly constrained linear programming

$$\min_{\boldsymbol{\beta}} \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \qquad \text{subject to } \boldsymbol{V}\boldsymbol{\beta} \leq \boldsymbol{v} \text{ and } \boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

- For example, MOSEK, LOQO, or SeDuMi.
- The software does not have to be sophisticated; just an elementary one is enough.