Pattern Information Processing: Constrained Least-Squares

Masashi Sugiyama
(Department of Computer Science)

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

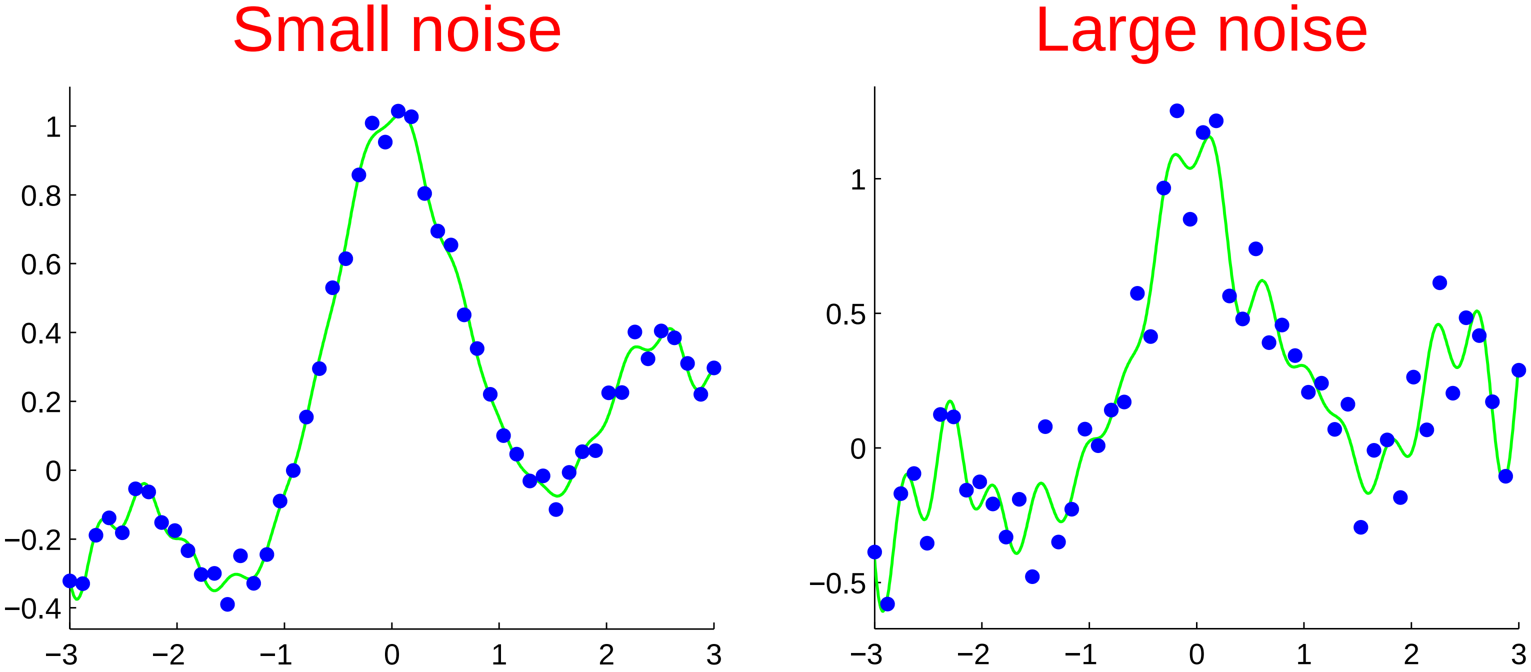
Over-fitting

- LS is proved to be a good learning method:
  - Unbiased and BLUE in realizable cases
  - Asymptotically unbiased and asymptotically efficient in unrealizable cases
- However, the learned function can over-fit to noisy examples (e.g., when the noise level is high).

Over-fitting

- Trigonometric polynomial model: $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$

$$\varphi_i(x) = \{1, \sin x, \cos x, \ldots, \sin 15x, \cos 15x\}$$



- In order to prevent over-fitting, model (search space) should be restricted appropriately.

Today's Plan

- Two approaches to restricting models:
  - Subspace LS
  - Quadratically constrained LS
- Sparseness and model choice.
- We focus on linear/kernel models.

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

Subspace LS

- Restrict the search space within a subspace

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

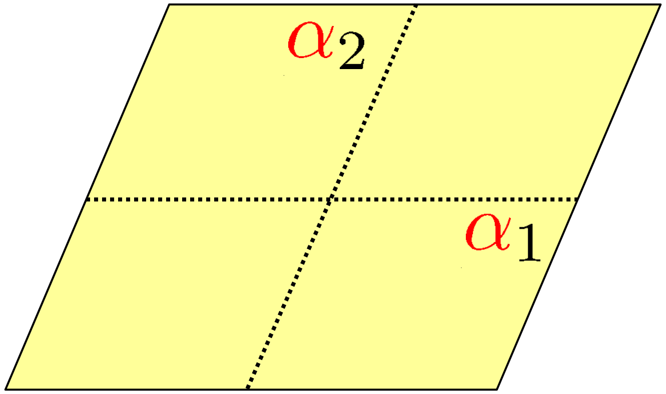
$$\text{subject to } \boldsymbol{P}\boldsymbol{\alpha} = \boldsymbol{\alpha}$$

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

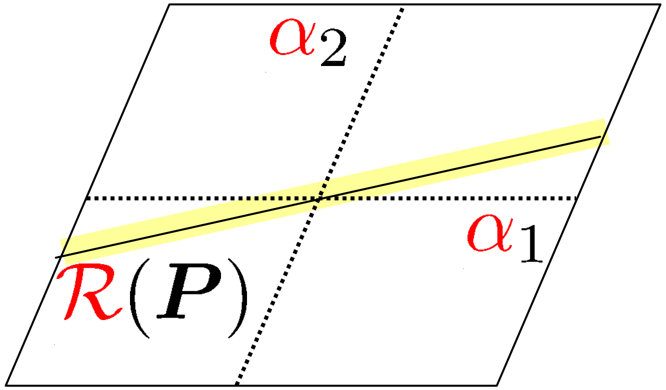
$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

$\boldsymbol{P}$: orthogonal projection onto the subspace

$$\boldsymbol{P}^2 = \boldsymbol{P} \qquad \boldsymbol{P}^\top = \boldsymbol{P}$$



Ordinary LS



Subspace LS

How to Obtain Solutions

- Since

$$J_{LS}(\boldsymbol{\alpha}) = \|\boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y}\|^2$$

just replacing $\boldsymbol{X}$ with $\boldsymbol{X}\boldsymbol{P}$ gives a solution:

$$\boldsymbol{L}_{SLS} = (\boldsymbol{P}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{P})^\dagger\boldsymbol{P}\boldsymbol{X}^\top$$

$$= (\boldsymbol{X}\boldsymbol{P})^\dagger$$

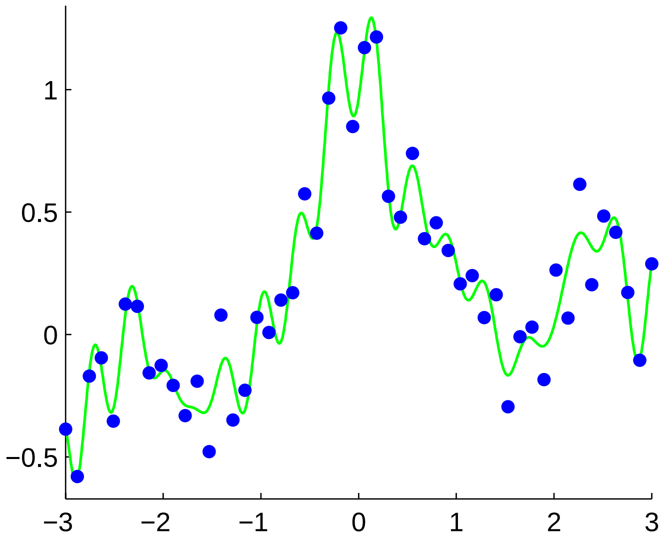
- $\dagger$:Moore-Penrose generalized inverse

$$\boldsymbol{B} = \boldsymbol{A}^\dagger \quad \Longleftrightarrow \quad \left\{ \begin{array}{l} \boldsymbol{A}\boldsymbol{B}\boldsymbol{A} = \boldsymbol{A} \\ \boldsymbol{B}\boldsymbol{A}\boldsymbol{B} = \boldsymbol{B} \\ (\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{A}\boldsymbol{B} \\ (\boldsymbol{B}\boldsymbol{A})^\top = \boldsymbol{B}\boldsymbol{A} \end{array} \right.$$

$$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}^\dagger = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/3 \end{pmatrix}$$

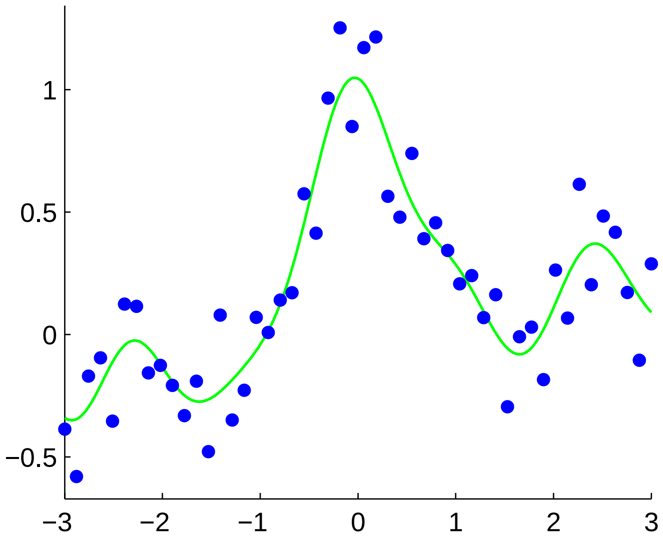
$$\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}^\dagger = \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix}$$

Example of SLS



Full LS

$1, \ldots, \sin 15x, \cos 15x$



Subspace LS

$1, \ldots, \sin 5x, \cos 5x$

- Over-fit can be avoided by properly choosing the subspace.

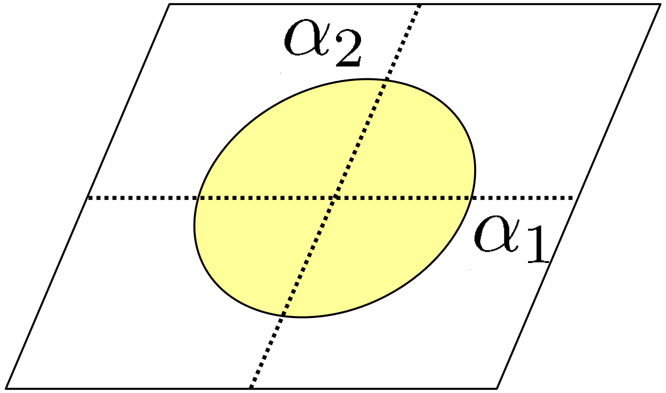
Quadratically Constrained LS

- Restrict the search space within a hyper-sphere.

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

$$\text{subject to } \|\boldsymbol{\alpha}\|^2 \leq C$$

$C \geq 0$



How to Obtain Solutions

- Lagrangian:

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

- λ : Lagrange multiplier
- Karush-Kuhn-Tucker (KKT) condition:
for some λ^*, the solution $\hat{\boldsymbol{\alpha}}_{QCLS}$ satisfies
  - $$\frac{\partial J_{QCLS}(\hat{\boldsymbol{\alpha}}_{QCLS},\lambda^*)}{\partial \boldsymbol{\alpha}} = \mathbf{0}$$
  - $\lambda^* \geq 0$
  - $\|\hat{\boldsymbol{\alpha}}_{QCLS}\|^2 - C \leq 0$
  - $\lambda^* \left(\|\hat{\boldsymbol{\alpha}}_{QCLS}\|^2 - C\right) = 0$

How to Obtain Solutions (cont.)

- $$\frac{\partial J_{QCLS}(\hat{\boldsymbol{\alpha}}_{QCLS}, \lambda^*)}{\partial \boldsymbol{\alpha}} = \mathbf{0}$$

⇒

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

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

- λ^* is obtained from $\lambda^* \left(\|\hat{\boldsymbol{\alpha}}_{QCLS}\|^2 - C \right) = 0$
- In practice, we start from $\lambda \ (\geq 0)$ and solve

$$\hat{\boldsymbol{\alpha}}_{QCLS} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^b}{\mathrm{argmin}}\, J_{QCLS}(\boldsymbol{\alpha})$$

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

Interpretation of QCLS

- QCLS tries to avoid overfitting by adding penalty (regularizer) to the “goodness-of-fit” term.

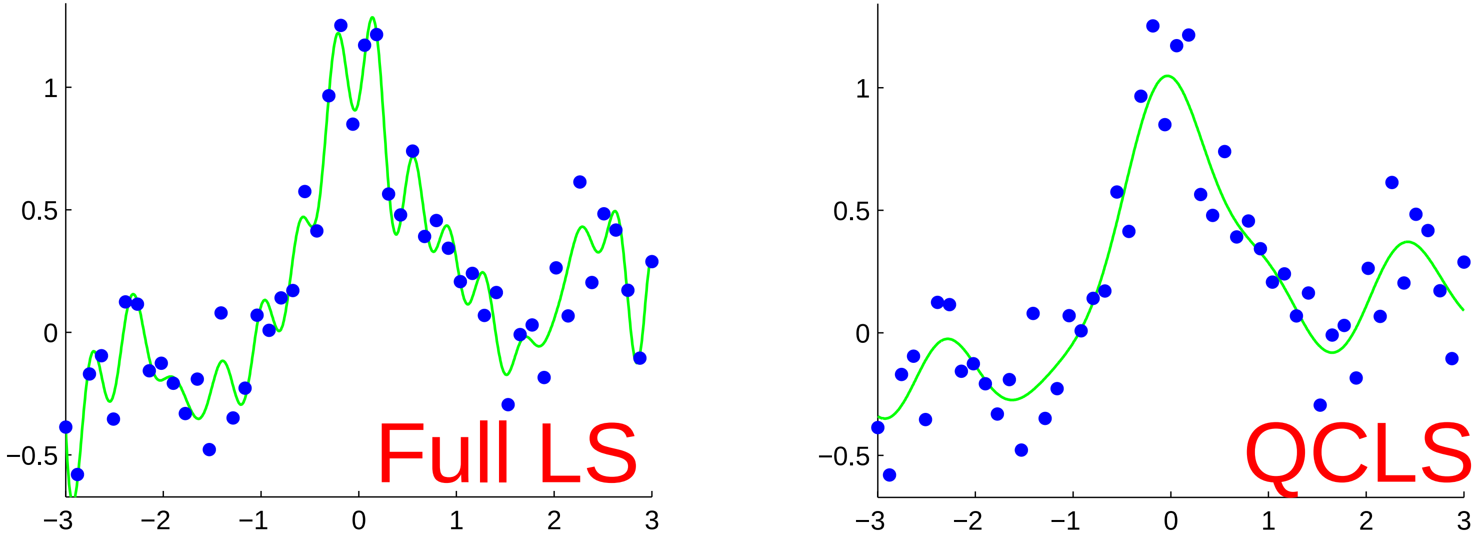
$$J_{QCLS}(\boldsymbol{\alpha}) = \underbrace{J_{LS}(\boldsymbol{\alpha})}_{\text{Good-ness of fit}} \quad + \quad \underbrace{\lambda\|\boldsymbol{\alpha}\|^2}_{\text{Penalty (regularizer)}} \quad + \quad \text{const.}$$

- For this reason, QCLS is also called quadratically regularized LS.
- λ is called the regularization parameter.

Example of QCLS

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

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



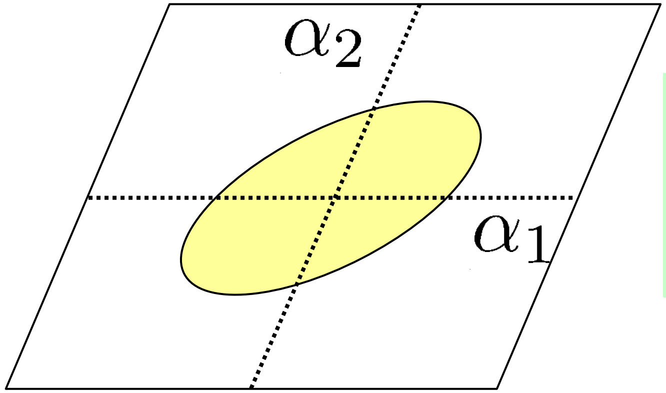
$(\lambda = 1)$

- Over-fit can be avoided by properly choosing the regularization parameter.

Generalization

- Restrict the search space within a hyper-ellipsoid.

$$\hat{\boldsymbol{\alpha}}_{QCLS} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^b}{\mathrm{argmin}}\, J_{LS}(\boldsymbol{\alpha}) \quad \text{subject to } \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \leq C$$



$C \geq 0$

$\boldsymbol{R}$:Positive semi-definite matrix ("regularization matrix")

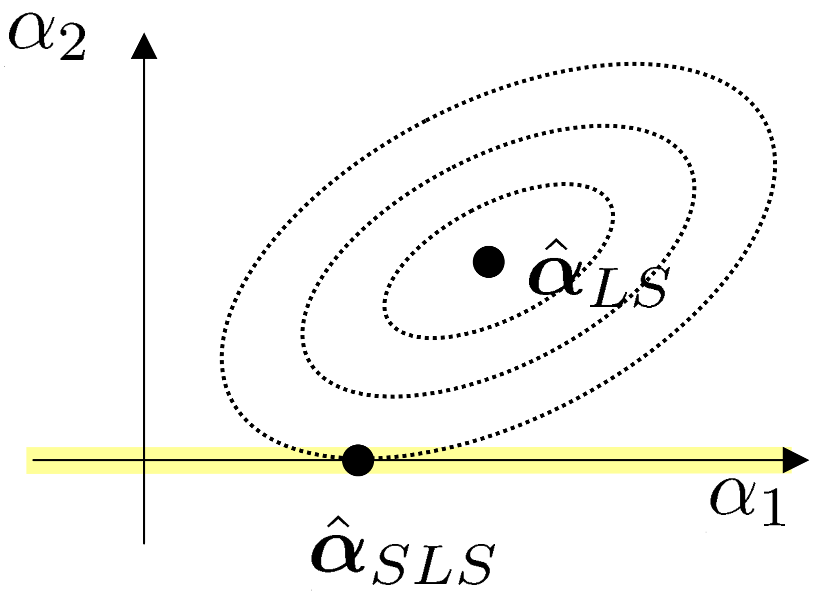
$\forall \boldsymbol{\alpha}, \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \geq 0$

- Solution: (proof is homework!)

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

Sparseness of Solution

- In SLS, if the subspace is spanned by a subset of basis functions $\{\varphi_i(\boldsymbol{x})\}_{i=1}^{b}$, some of the parameters $\{\alpha_i\}_{i=1}^{b}$ are exactly zero.


α_2
$\hat{\boldsymbol{\alpha}}_{LS}$
α_1
$\hat{\boldsymbol{\alpha}}_{SLS}$

Model Choice

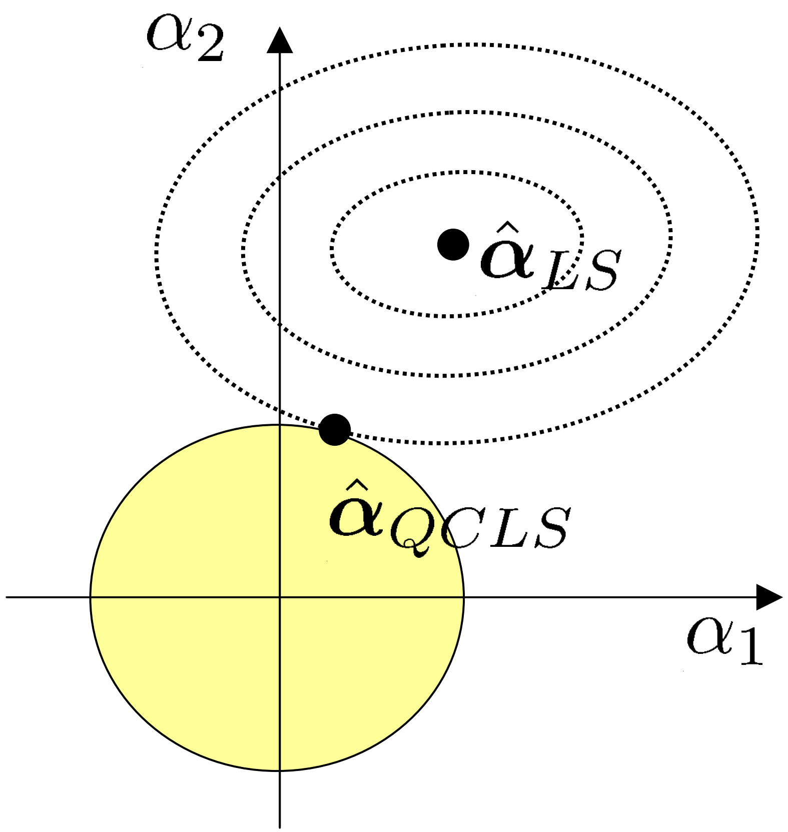
- Sparse solution is computationally advantageous when calculating the output values.

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

- However, the possible choices of such subspaces are combinatorial: 2^b
- Computationally infeasible to find the best subset.

Property of QCLS

- In QCLS, model choice is continuous: λ
- However, solution is not generally sparse.


α_2
$\hat{\boldsymbol{\alpha}}_{LS}$
$\hat{\boldsymbol{\alpha}}_{QCLS}$
α_1

Homework

1. Prove that the solution of

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

$$\text{subject to } \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \leq C$$

is given by

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

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

Homework (cont.)

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

   and analyze the results, e.g., changing
   - Target functions
   - Number of samples
   - Noise level
   - Width of Gaussian kernel
   - Regularization parameter/matrix

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

$$\min_{\boldsymbol{\beta}} \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle$$

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

- For example, MOSEK, LOQO, or SeDuMi.
- The software is not necessarily sophisticated; just elementary one is enough.