Pattern Information Processing:⁶⁵ Sparse Methods

Masashi Sugiyama (Department of Computer Science)

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

Sparseness and Continuous Model Choice

Two approaches to avoiding over-fitting:

	Sparseness	Model parameter
Subspace LS	Yes	Discrete
Quadratically constrained LS	No	Continuous

We want to have sparseness and continuous model choice at the same time.

Today's Plan

Sparse learning method

- How to deal with absolute values in optimization
- Standard form of quadratic programs

Non-Linear Learning for Linear / Kernel Models

Linear / kernel models

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

Non-linear learning

$$\hat{oldsymbol{lpha}} = oldsymbol{L}(oldsymbol{y})$$

L:Non-linear function

68

I1-Constrained LS

Restrict the search space within a hyper-rhombus.

$$\hat{\boldsymbol{\alpha}}_{\ell_1 CLS} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^b}{\operatorname{subject}} \quad \text{subject to } \|\boldsymbol{\alpha}\|_1 \leq C$$
$$\ell_1 - \operatorname{norm}$$
$$\|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^b |\alpha_i|$$

See:

Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society, Series B, 58(1), 267-288,1996. Chen, Donoho & Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing, 20(1), 33-61, 1998.

Why Sparse?

The solution is often exactly on an axis.



How to Obtain Solutions

Lagrangian: J_{ℓ1}CLS(α) = J_{LS}(α) + λ||α||₁
λ :Lagrange multiplier
Similar to QCLS, we practically start from λ (≥ 0) and solve
âℓ CLS = argmin Je CLS(α)

$$\hat{\boldsymbol{\alpha}}_{\ell_1 CLS} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^b} J_{\ell_1 CLS}(\boldsymbol{\alpha})$$

It is often called ℓ_1 regularized LS.

How to Obtain Solutions (cont.)⁷²
How to deal with l₁-norm?
Use the following identity:

 $egin{aligned} \|oldsymbollpha\|_1 &= \min_{oldsymbol u \in \mathbb{R}^b} \sum_{i=1}^b u_i \ & ext{subject to } -oldsymbol u \leq oldsymbollpha \leq oldsymbol u, \end{aligned}$

Note: Inequality in constraint is component-wise

Intuition: Obtain smallest box that includes



How to Obtain Solutions (cont.)⁷³
Proof: Let
$$\hat{u} = \underset{u \in \mathbb{R}^b}{\operatorname{argmin}} \sum_{i=1}^{b} u_i$$

subject to $-u < \alpha < u$,

The constraint implies $\hat{u}_i \ge |\alpha_i|$. Suppose $\hat{u}_i > |\alpha_i|$. Then such \hat{u}_i is not a solution since $\tilde{u}_i = |\alpha_i|$ gives a smaller value: $\sum_{i=1}^{b} \tilde{u}_i < \sum_{i=1}^{b} \hat{u}_i$

This implies that the solution satisfies $\hat{u}_i = |\alpha_i|$, which yields $\sum_{i=1}^{b} \hat{u}_i = \sum_{i=1}^{b} |\alpha_i| = ||\boldsymbol{\alpha}||_1$

How to Obtain Solutions (cont.)⁷⁴

$$\hat{\boldsymbol{\alpha}}_{\ell_1 CLS} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^b} J_{\ell_1 CLS}(\boldsymbol{\alpha})$$
$$J_{\ell_1 CLS}(\boldsymbol{\alpha}) = J_{LS}(\boldsymbol{\alpha}) + \lambda \|\boldsymbol{\alpha}\|_1$$

$\hat{\alpha}_{\ell_1 CLS}$ is given as the solution of

$$\min_{\boldsymbol{\alpha},\boldsymbol{u}\in\mathbb{R}^b} \left[J_{LS}(\boldsymbol{\alpha}) + \lambda \sum_{i=1}^b u_i \right]$$

subject to $-u \leq \alpha \leq u$,

Linearly Constrained Quadratic⁷⁵ Programming Problem

Standard optimization software can solve the following form of linearly constrained quadratic programming problems.

$$\begin{split} \min_{\boldsymbol{\beta}} \left[\frac{1}{2} \langle \boldsymbol{Q} \boldsymbol{\beta}, \boldsymbol{\beta} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \right] \\ \text{subject to } \boldsymbol{V} \boldsymbol{\beta} \leq \boldsymbol{v} \\ \boldsymbol{G} \boldsymbol{\beta} = \boldsymbol{g} \end{split}$$

Standard Form

 ℓ_1 -constrained LS can be expressed as

$$egin{array}{rcl} eta &=& \left(egin{array}{c} lpha \ u \end{array}
ight) \ egin{array}{rcl} eta &=& 2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} X \Gamma_{oldsymbol lpha} \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& -2 \Gamma_{oldsymbol lpha}^{ op} X^{ op} y + \lambda \Gamma_{oldsymbol u}^{ op} \mathbf{1}_p \ eta &=& 0_{2p}, \ eta &=& 0_{2p}, \ eta &=& 0_{2p}, \ ela &=& 0_{2p}, \ ela &=& 0_{2p}. \ ela &=& 0_{2p}. \end{array}$$

Example of Sparse Learning
 Gaussian kernel model:



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

27 out of 50 parameters are exactly zero.

Constrained LS

	Sparseness	Model parameter	Parameter learning
Subspace LS	Yes	Discrete	Analytic (Linear)
Quadratically constrained LS	No	Continuous	Analytic (Linear)
ℓ_1 constrained LS	Yes	Continuous	Iterative (Non-linear)

Homework

1. Derive the standard quadratic programming form of l_1 -constrained LS by yourself.

Homework (cont.)

- 2. For your own toy 1-dimensional data, perform simulations using
 - Gaussian kernel models
 - ℓ_1 -constraint least-squares learning

and analyze the results, e.g., by changing

- Target functions
- Number of samples
- Noise level
- Width of Gaussian kernel
- Regularization parameter