Feature Selection and Sparsity

Makoto Yamada
myamada@i.kyoto-u.ac.jp

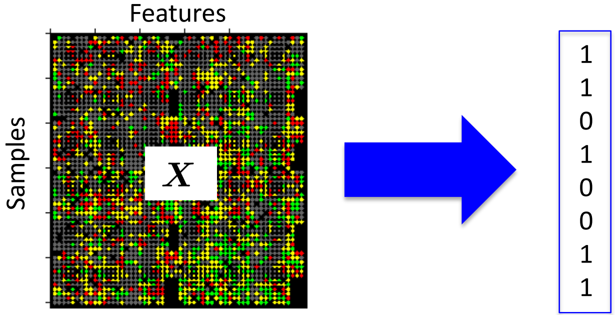
Kyoto University

June/4/2018

Introduction

Feature selection is important for handling high-dimensional data:

- User data ($d > 100$)
  e.g., e-mail spam detection.
- Gene expression data ($d > 20000$)
  e.g., cancer classification.
- Text based feature such as TF-IDF ($d > 100,000$)
  e.g., Sentiment analysis



Motivation1

The purpose of feature selection is

- to improve the prediction accuracy by getting rid of non-important features.
- to make the prediction faster.
- to interpret data.
- to handle high-dimensional data.

Motivation2

Let us think about a least-squared regression problems:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^\top \in \mathbb{R}^d$ and $\boldsymbol{y} \in \mathbb{R}^n$, and $\| \cdot \|_2^2$ is the ℓ_2 norm.

Question:

- $d < n$ and the rank of $\boldsymbol{X}$ is d. Please derive the analytical solution of $\boldsymbol{w}$.

Motivation2

Take the objective function with respect to $\boldsymbol{w}$ and set it to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 = -2\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}) = \boldsymbol{0}$$

Use Eq. (84) of [1]. The solution is given as

$$\widehat{\boldsymbol{w}} = (\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{X}\boldsymbol{y}.$$

If the rank of $\boldsymbol{X}$ is d, the rank of $\boldsymbol{X}\boldsymbol{X}^\top$ is also d and it is invertible.

What happens if the rank of $\boldsymbol{X}$ is less than d?

- $\boldsymbol{X}\boldsymbol{X}^\top$ is not invertible.
- Maybe, we can add a regularizer (or use pseudo-inverse). we get a dense solution and numerically unstable :(

A possible solution is to use feature selection! If we select $r < d$ features, we can compute $\boldsymbol{w}$.

Problem formulation

Problem formulation of feature selection

- Input vector: $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^\top \in \mathbb{R}^d$
- Output: $y \in \mathbb{R}$
- Paired data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

Goal: Select $r (r < d)$ features of input $\boldsymbol{x}$ that are responsible for output y.

Problems: There is 2^d combinations :(It is hard even if d is 100.

Feature Selection Algorithms

The feature selection algorithms are categorized into three types:

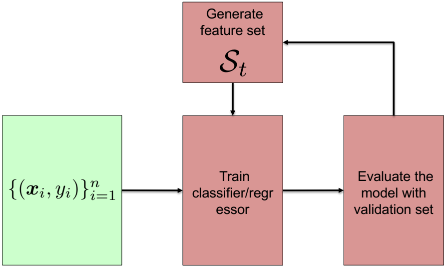
- Wrapper Method
  Use a predictive model to select features.
- Filter Method
  Use a proxy measure (such as mutual information) instead of the error rate to select features.
- Embedded Method
  Features are selected as part of the model construction process.

Wrapper Method

Use a predictive model (e.g., classifier) to select features.

The simplest approach would be...

1. Generate feature set $\mathcal{S}_t$
2. Train predictive model with $\mathcal{S}_t$ and test the prediction accuracy with hold-out set.
3. Iterate 1 and 2 until all feature combination is examined.



Wrapper Method

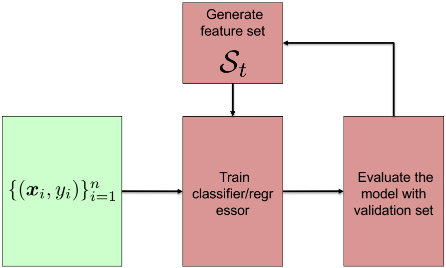
Pro:

- It can select features that have feature-feature interaction.

Cons:

- It can be overfitted if the number of samples is insufficient.
- Computationally expensive.

Wrapper method is not that popular compared to filter and embedded methods...



Filter Method

Use a proxy measure (such as mutual information) instead of the error rate to select features.

Pros:

- Easy to implement.
- It scales well (easy to implement with distributed computing).
- Can select features from high-dimensional data (both linear and nonlinear way).

Cons:

- The feature selection is independent of the model. The selected features may not be the best set to achieve highest accuracy.
- It is hard to detect select features with interaction. (Of course, we can somehow select them, but it increase computation cost.

Filter Method (Example)

Maximum Relevance Feature Selection (MR)

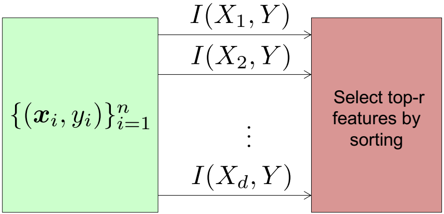
Compute association score between each feature and its output and rank them.

- Correlation, Mutual information, and the kernel based independence measures are used.
- Easy to implement and it scales well.

Optimization problem:

$$\max_{\boldsymbol{\beta} \in \{0,1\}^d} \quad \frac{1}{S} \sum_{k=1}^{d} \beta_k I(X_k, Y),$$

where $S = \beta_1 + \ldots + \beta_d$.



Filter Method (Example)

Minimum Redundancy Maximum Relevance (mRMR) [2]

MR feature selection tends to select redundant features.

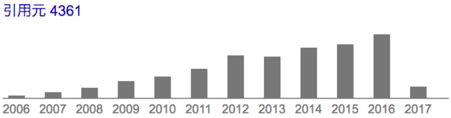
mRMR method is to

- select features that have high association to its output.
- select independent features.

Optimization problem:

$$\max_{\boldsymbol{\beta}\in\{0,1\}^d} \quad \frac{1}{S}\sum_{k=1}^{d}\beta_k I(X_k, Y) - \frac{1}{S^2}\sum_{k=1}^{d}\sum_{k'=1}^{d}\beta_k\beta_{k'} I(X_k, X_{k'}).$$

This optimization problem can be solved by using greedy algorithm.



Filter Method (Mutual Information)

To optimize mRMR, we tend to use the mutual information as an association score.

Independence:

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})$$

Mutual Information:

$$\mathrm{MI}(X, Y) = \iint p(\boldsymbol{x}, \boldsymbol{y}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}$$

Under independence:

$$\mathrm{MI}(X, Y) = \iint p(\boldsymbol{x}, \boldsymbol{y}) \log \frac{p(\boldsymbol{x})p(\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y} = 0$$

Filter Method (Example)

How to optimize mRMR? Suppose we use the mutual information as a dependency measure and we already have $\mathcal{S}_{m-1}$, which is the feature set with $m-1$ features, then we can select the m-th feature by solving the following optimization problem:

$$\max_{j \in \bar{\mathcal{S}}_{m-1}} \mathrm{MI}(X_j, Y) - \frac{1}{m-1} \sum_{i \in \mathcal{S}_{m-1}} \mathrm{MI}(X_j, X_i). \tag{1}$$

We select a feature that having high dependency with Y and independent of features in $\mathcal{S}_{m-1}$.

Filter Method (Continuous optimization)

The MR and mRMR feature selection algorithms are discrete optimization problem. In feature selection, continuous optimization based approach is also popular.

The key idea is to relax the condition (i.e., allow to take continuous number).

Quadratic Programming Feature Selection [3]:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^d} \quad \sum_{k=1}^{d}\alpha_k I(X_k, Y) - \frac{1}{2}\sum_{k=1}^{d}\sum_{k'=1}^{d}\alpha_k\alpha_{k'} I(X_k, X_{k'}),$$
$$\text{s.t.} \quad \alpha_1 + \alpha_2 + \ldots + \alpha_d = 1,\ \alpha_1, \ldots, \alpha_d \geq 0$$

Filter Method (Continuous optimization)

Let us denote:

$$\boldsymbol{h}_j = I(X_j, Y),$$
$$\boldsymbol{H}_{ij} = I(X_k, X_{k'})$$

where $\boldsymbol{h} \in \mathbb{R}^d$ and $\boldsymbol{H} \in \mathbb{R}^{d \times d}$.
We have

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^d} \quad \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{H}\boldsymbol{\alpha} - \boldsymbol{h}^\top \boldsymbol{\alpha}$$
$$\text{s.t.} \quad \boldsymbol{\alpha}^\top \mathbf{1} = 1$$

This is a quadratic programming with simplex constraint (can be solved by using an off-the-shelf package).

Note: For mutual information, $\boldsymbol{H}$ may not be positive definite. It can be non-convex optimization.

Embedded Method

Features are selected as part of the model construction process.

Embedded method can be regarded as an intermediate method between wrapper and filter methods.

Pros:

- Can select features with high prediction accuracy.
- Computationally efficient than wrapper method.

Cons:

- Computationally expensive than filter method.
- If the input output relationship are nonlinear, it is computationally expensive. It is more suited for linear method.

Embedded Method (Lasso)

Least Absolute Shrinkage and Selection Operator (Lasso)

The optimization problem of Lasso can be written as

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1,$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ is the input matrix and $\boldsymbol{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ is the output vector.

$$\|\boldsymbol{w}\|_1 = \sum_{k=1}^{d} |w_k|$$

is an ℓ_1 norm.

Lasso is a convex method: The first term is a convex function w.r.t. $\boldsymbol{w}$. ℓ_1 norm (all norm) is convex:

$$\begin{aligned}\|\alpha\boldsymbol{w} + (1-\alpha)\boldsymbol{v}\|_1 &\leq \|\alpha\boldsymbol{w}\|_1 + \|(1-\alpha)\boldsymbol{v}\|_1 \quad (\textit{triangle inequality}) \\ &= \alpha\|\boldsymbol{w}\|_1 + (1-\alpha)\|\boldsymbol{v}\|_1 \quad (\textit{absolutely scalable}),\end{aligned}$$

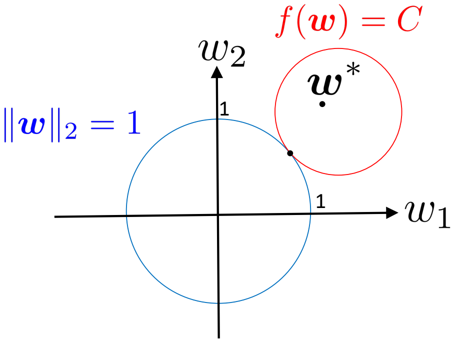
where $0 \leq \alpha \leq 1$. The sum of two convex functions is convex.

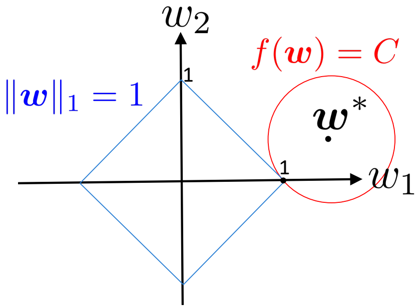
Embedded Method (Lasso) Some intuitive explanation

Using the ℓ_1 regularizer, we can make $\boldsymbol{w}$ sparse.
The ℓ_1 regularization is equivalent to ℓ_1 norm constraint:

$$\min_{\boldsymbol{w}} \ f(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_1 \longrightarrow \min_{\boldsymbol{w}} \ f(\boldsymbol{w}), \ \text{s.t.} \ \|\boldsymbol{w}\|_1 \leq \eta.$$

If we consider the Lagrange function of the ℓ_1 norm constraint, there exists the same solution of the ℓ_1 norm constraint with an arbitrary λ.
Level curves of norms and loss:





Embedded Method (Lasso)

Lasso has no closed form solution. Thus, we need to iteratively optimize the problem.

Here, we introduce the Alternating Direction Method of Multipliers (ADMM) [4].

We can rewrite the Lasso optimization problem as

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{z}} \quad & \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{z}\|_1 + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{w} = \boldsymbol{z} \end{aligned}$$

The key idea here is to split the main objective and the non-differentiable regularization term. Since the last term $\frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2$ is zero if the constraint is satisfied, this problem is equivalent to the original Lasso problem.

Embedded Method (Lasso)

Let us denote the Lagrange multipliers as $\boldsymbol{\gamma} \in \mathbb{R}^d$, we can write a Lagrangian function (called Augmented Lagrangian function) as follows:

$$J(\boldsymbol{w}, \boldsymbol{z}, \gamma) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 + \gamma^\top(\boldsymbol{w} - \boldsymbol{z}) + \lambda\|\boldsymbol{z}\|_1 + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2,$$

where $\rho > 0$ is a tuning parameter.
In ADMM, we consider the following optimization problem:

$$\max_{\boldsymbol{\gamma}} \min_{\boldsymbol{w},\boldsymbol{z}} \quad J(\boldsymbol{w}, \boldsymbol{z}, \gamma) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 + \gamma^\top(\boldsymbol{w} - \boldsymbol{z}) + \lambda\|\boldsymbol{z}\|_1 + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{z}\|_2^2,$$

Since we have the relationship,

$$\max_{\boldsymbol{\gamma}} J(\boldsymbol{w}, \boldsymbol{z}, \gamma) = \begin{cases} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{z}\|_1 & (\boldsymbol{w} = \boldsymbol{z}) \\ \infty & (\text{Otherwise}) \end{cases}$$

The optimization problem is equivalent to the original Lasso problem.

Embedded Method (Lasso)

Minimizing $J(\boldsymbol{w}, \boldsymbol{z}, \gamma)$ w.r.t. $\boldsymbol{w}$. If we fix $\boldsymbol{z}$ and γ as $\boldsymbol{z}^{(t)}$ and $\gamma^{(t)}$, $J(\boldsymbol{w}, \boldsymbol{z}^{(t)}, \gamma^{(t)})$ is convex w.r.t. $\boldsymbol{w}$. That is,

$$\frac{\partial J(\boldsymbol{w}, \boldsymbol{z}, \gamma)}{\partial \boldsymbol{w}} = -\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}) + \gamma + \rho(\boldsymbol{w} - \boldsymbol{z}) = \boldsymbol{0}.$$

Here, we can use the following equation (see [1] Eq. (84)):

$$\frac{\partial \|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2}{\partial \boldsymbol{w}} = -2\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}).$$

Solving it for $\boldsymbol{w}$:

$$\begin{aligned}(\boldsymbol{X}\boldsymbol{X}^\top + \rho \boldsymbol{I})\boldsymbol{w} &= \boldsymbol{X}\boldsymbol{y} - \gamma^{(t)} + \rho \boldsymbol{z}^{(t)} \\ \boldsymbol{w}^{(t+1)} &= (\boldsymbol{X}\boldsymbol{X}^\top + \rho \boldsymbol{I})^{-1}(\boldsymbol{X}\boldsymbol{y} - \gamma^{(t)} + \rho \boldsymbol{z}^{(t)}).\end{aligned}$$

Embedded Method (Lasso)

Minimizing $J(\boldsymbol{w}, \boldsymbol{z}, \gamma)$ w.r.t. $\boldsymbol{z}$. If we fix $\boldsymbol{w}$ and γ as $\boldsymbol{w}^{(t)}$ and $\gamma^{(t)}$, $J(\boldsymbol{w}^{(t)}, \boldsymbol{z}, \gamma^{(t)})$ is convex w.r.t. $\boldsymbol{z}$.

$$J(\boldsymbol{w}^{(t)}, \boldsymbol{z}, \gamma^{(t)}) = \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{w}^{(t)}\|_2^2 + \lambda\|\boldsymbol{z}\|_1 - \boldsymbol{\gamma}^\top \boldsymbol{z} + \text{Const.}$$

$\|\boldsymbol{z}\|_1$ is not differentiable at 0. However, we can analytically solve the problem! Moreover, since there is no interaction in the elements of $\boldsymbol{z}$, we can solve it for each element.

$$J(\boldsymbol{w}^{(t)}, [z_1, \ldots, z_\ell, \ldots, z_d], \gamma^{(t)}) = \frac{\rho}{2}(z_\ell - w_\ell^{(t)})^2 + \lambda|z_\ell| - \gamma_\ell z_\ell + \text{Const.}$$

Embedded Method (Lasso)

$$J(\boldsymbol{w}^{(t)}, [z_1, \ldots, z_\ell, \ldots, z_d], \boldsymbol{\gamma}^{(t)}) = \frac{\rho}{2}(z_\ell - w_\ell^{(t)})^2 + \lambda|z_\ell| - \gamma_\ell z_\ell + \text{Const.}$$

Case1: $z_\ell > 0, \rho(z_\ell - w_\ell^{(t)}) + \lambda - \gamma_\ell = 0 \longrightarrow z_\ell = w_\ell^{(t)} + \frac{1}{\rho}(\gamma_\ell - \lambda)$

That is, $z_\ell > 0$ if $w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell > \frac{\lambda}{\rho}$

Case2: $z_\ell < 0, \rho(z_\ell - w_\ell^{(t)}) - \lambda - \gamma_\ell = 0 \longrightarrow z_\ell = w_\ell^{(t)} + \frac{1}{\rho}(\gamma_\ell + \lambda)$

That is, $z_\ell < 0$ if $w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell < -\frac{\lambda}{\rho}$

Case3: $z_\ell = 0$,
$0 \in \rho(z_\ell - w_\ell^{(t)}) + \lambda[-1\ 1] - \gamma_\ell \longrightarrow w_\ell + \frac{1}{\rho}\gamma_\ell \in [-\frac{\lambda}{\rho}, \frac{\lambda}{\rho}], (z_\ell = 0)$.

Therefore, we have

$$z_\ell = \begin{cases} w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell - \frac{\lambda}{\rho} & (w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell > \frac{\lambda}{\rho}) \\ 0 & (w_\ell + \frac{1}{\rho}\gamma_\ell \in [-\frac{\lambda}{\rho}, \frac{\lambda}{\rho}]) \\ w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell + \frac{\lambda}{\rho} & (w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell < -\frac{\lambda}{\rho}) \end{cases}$$

Embedded Method (Lasso)

Let us introduce the Soft-Thresholding function: (Figure)

$$S_\lambda(x) = \begin{cases} x - \lambda & (x > \lambda) \\ 0 & (x \in [-\lambda, \lambda]) \\ x + \lambda & (x < -\lambda) \end{cases},$$
$$= \mathrm{sign}(x)\max(0, |x| - \lambda)$$

Therefore, the update of z_ℓ can be simply written by the soft-thresholding function as

$$\widehat{z}_\ell^{(t+1)} = S_{\frac{\lambda}{\rho}}(w_\ell^{(t)} + \frac{1}{\rho}\gamma_\ell).$$

Embedded Method (Lasso)

Maximizing $J(\boldsymbol{w}, \boldsymbol{z}, \gamma)$ w.r.t. γ. That is the optimization problem can be written as

$$\max_{\boldsymbol{\gamma}} J(\boldsymbol{w}, \boldsymbol{z}, \gamma) = \gamma^\top(\boldsymbol{w} - \boldsymbol{z}).$$

To optimize this problem, since we cannot get the analytical solution, we use the gradient ascent algorithm:

$$\gamma^{(t+1)} = \gamma^{(t)} + \rho(\boldsymbol{w}^{(t)} - \boldsymbol{z}^{(t)}).$$

Thus, the ADMM algorithm for Lasso can be summarized as

$$\begin{aligned}
\boldsymbol{w}^{(t+1)} &= (\boldsymbol{X}\boldsymbol{X}^\top + \rho\boldsymbol{I})^{-1}(\boldsymbol{X}\boldsymbol{y} - \gamma^{(t)} + \rho\boldsymbol{z}^{(t)}) \\
\boldsymbol{z}_\ell^{(t+1)} &= S_{\frac{\lambda}{\rho}}(\boldsymbol{w}^{(t+1)} + \frac{1}{\rho}\gamma) \\
\gamma^{(t+1)} &= \gamma^{(t+1)} + \rho(\boldsymbol{w}^{(t+1)} - \boldsymbol{z}^{(t)}).
\end{aligned}$$

Kaare Brandt Petersen, Michael Syskind Pedersen, et al.
The matrix cookbook.
Technical University of Denmark, 7:15, 2008.

H. Peng, F. Long, and C. Ding.
Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 27:1226–1237, 2005.

I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz.
Quadratic programming feature selection.
JMLR, 11:1491–1516, 2010.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al.
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Foundations and Trends® in Machine learning, 3(1):1–122, 2011.