Gradient LASSO algoithm

Yongdai Kim

Seoul National University, Korea

jointly with

Yuwon Kim

University of Minnesota, USA

and

Jinseog Kim

Statistical Research Center for Complex Systems, Korea

Contents

1. **Introduction**

- LASSO (Least Absolute Shrinkage and Selection Operator, Tibshirani, 1996) achieves better prediction accuracy by shrinkage, and at the same time it gives a sparse solution (some coefficients are exactly 0).
- A problem in LASSO is that the objective function is not differentiable, and hence special optimization techniques (QP or non-linear programing) are necessary.
- In this talk, I propose a gradient descent algorithm for LASSO, called *gradient LASSO*, which is simpler and stabler than the existing algorithms and can be applied to very large dimensional data easily.

2. **LASSO: Review**

- Covariate : $\mathbf{x} = (x_1, \ldots, x_p)^{'} \in R^p$
- Response : $y \in R$
- Model: $y = \boldsymbol{\beta}^{'}\mathbf{x} + \epsilon$ or $\Pr(y = 1|\mathbf{x}) = \text{logit}(\boldsymbol{\beta}^{'}\mathbf{x})$ or $\ldots$
- Data: $\{(y_i, \mathbf{x}_i) \;\; i = 1, \ldots, n\}$
- Objective: Estimate $\boldsymbol{\beta}$ using the data.
- Question: How do we estimate $\boldsymbol{\beta}$ when p is large compared to n?
- Two methods
  - (variable) Selection
  - Shrinkage

Selection

- Select a subset of $\{x_1, \ldots, x_p\}$ and use only the selected covariates for model fitting.
- Popular methods for selection
  - All possible
  - Forward Selection
  - Backward elimination
  - Stepwise
- The predictive model is unstable due to the nature of discreteness of the selection process, which may result in inferior prediction error.

Shrinkage: Ridge regression

- Estimate $\boldsymbol{\beta}$ by minimizing

$$\sum_{i=1}^{n} l(y_i, \boldsymbol{\beta}' \mathbf{x}_i) + \lambda \sum_{j=1}^{p} \beta_j^2$$

  where $\lambda > 0$ and l is a loss function.

- Why "Shrinkage" ? : It can be shown that $\|\hat{\boldsymbol{\beta}}^{Ridge}\|_2 \leq \|\hat{\boldsymbol{\beta}}\|_2$ where $\hat{\boldsymbol{\beta}}^{Ridge}$ is the Ridge estimator and $\hat{\boldsymbol{\beta}}$ is the ordinary estimator (Ridge estimator with $\lambda = 0$).

- The Ridge estimator is stabler than that from the selection and so gives better accuracy.

- A disadvantage of Ridge regression is that the interpretation is hard (i.e. none of the estimated coefficients is 0).

Comparison of Selection and Shrinkage

Term	LS	Selection	Ridge
Intercept	2.480	2.495	2.467
x_1	0.680	0.740	0.389
x_2	0.305	0.367	0.238
x_3	-0.141		-0.029
x_4	0.210		0.159
x_5	0.305		0.217
x_6	-0.288		0.026
x_7	-0.021		0.042
x_8	0.267		0.123
Test Error	0.586	0.574	0.540

Can we do selection and shrinkage simultaneously? Yes, it is LASSO!!!

LASSO

- LASSO estimates $\boldsymbol{\beta}$ by minimizing

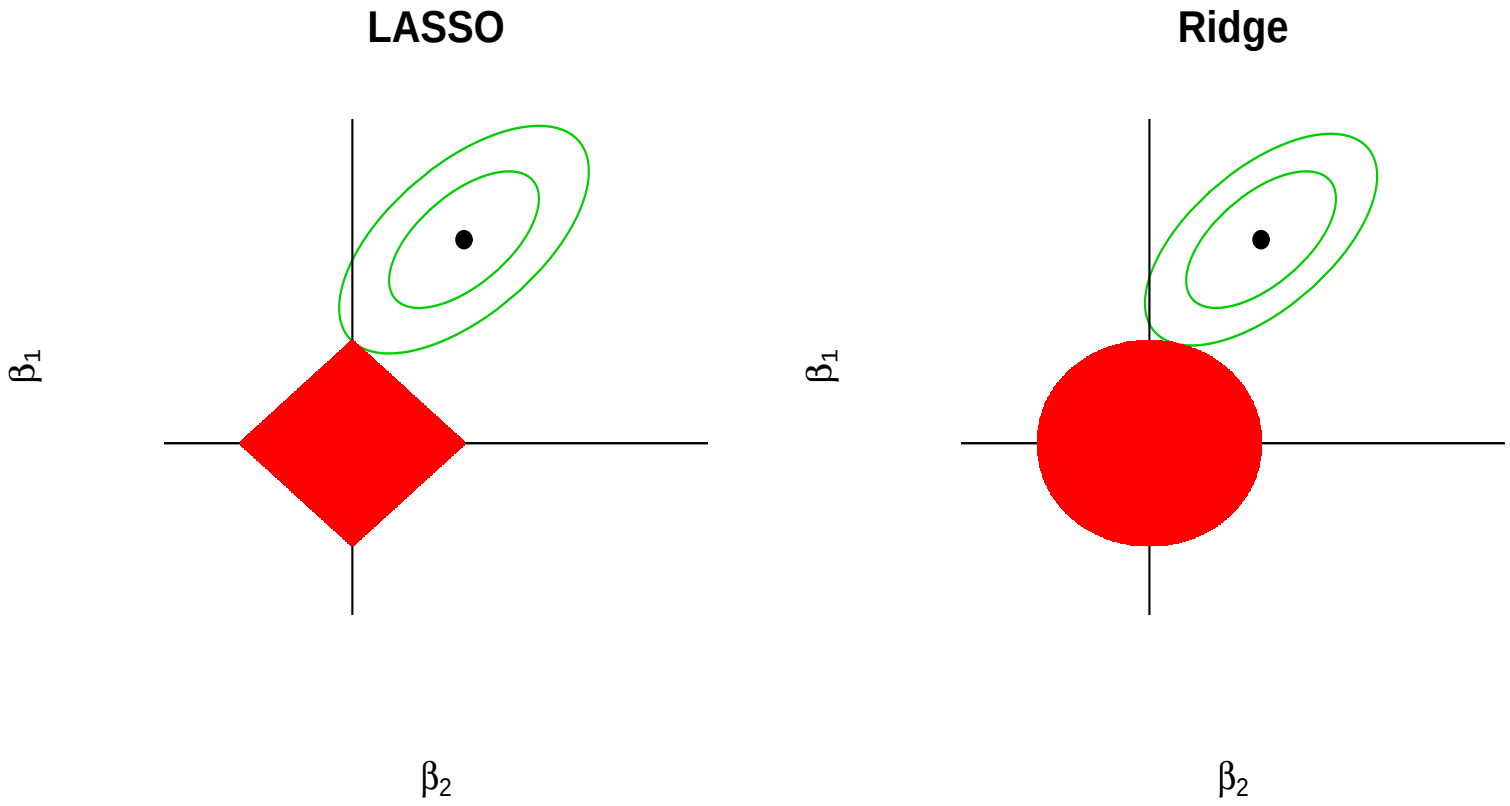
$$\sum_{i=1}^{n} l(y_i, \boldsymbol{\beta}^{'} \mathbf{x}_i) + \lambda \sum_{j=1}^{p} |\beta_j|.$$

- LASSO is first proposed by Tibshirani (1996).
- One very interesting property of LASSO is that the predictive model is sparse (i.e. some coefficients are exactly 0).

Comparison of LASSO with the other methods

Term	LS	Selection	Ridge	LASSO
Intercept	2.480	2.495	2.467	2.477
x_1	0.680	0.740	0.389	0.545
x_2	0.305	0.367	0.238	0.237
x_3	-0.141		-0.029	
x_4	0.210		0.159	0.098
x_5	0.305		0.217	0.165
x_6	-0.288		0.026	
x_7	-0.021		0.042	
x_8	0.267		0.123	0.059
Test Error	0.586	0.574	0.540	0.491

Why is LASSO sparse?



3. **Algorithms: Review**

- As mentioned in Introduction, LASSO is computationally demanding since the L_1 constraint is not differentiable.
- There are various special algorithms for LASSO:
  - Osborne's algorithm
  - Grafting
  - ϵ-boosting
- Let $R(\boldsymbol{\beta}) = \sum_{i=1}^{n} l(y_i, \boldsymbol{\beta}^{'} \mathbf{x}_i)$ be the empirical risk.
- Assume R is convex and differentiable (i.e. smooth convex loss).

Osborne's algorithm (Osborne et al. 2000)

- Implemented in R.
- The Obsnorne's algorithm repeatedly updates the solution through (i) optimization, (ii) deletion and (iii) optimality check and addition.
- For a current solution $\boldsymbol{\beta}$, let $\sigma = \{j : \beta_j \neq 0\}$.

- Optimization
  - $H_\sigma = \{\mathbf{h} \in R^p : h_k = 0 \text{ if } k \notin \sigma\}$.
  - Solve

    $$\text{Minimize}_{\mathbf{h}_\sigma \in H_\sigma} R(\boldsymbol{\beta} + \mathbf{h}_\sigma)$$

    subject to $\text{sign}(\boldsymbol{\beta})^{'}(\boldsymbol{\beta} + \mathbf{h}_\sigma) \leq \lambda$.
  - If $\text{sign}(\boldsymbol{\beta} + \mathbf{h}_\sigma) = \text{sign}(\boldsymbol{\beta})$, then $\boldsymbol{\beta} + \mathbf{h}_\sigma$ is feasible (i.e. $\|\boldsymbol{\beta} + \mathbf{h}_\sigma\|_1 \leq \lambda$), then we update $\boldsymbol{\beta} = \boldsymbol{\beta} + \mathbf{h}_\sigma$ and go to the optimality check step. Otherwise, we go to the deletion step.
  - Note: With the square error loss, $\mathbf{h}_\sigma$ has the closed form solution which includes $(\mathbf{X}_\sigma^{'}\mathbf{X}_\sigma)^{-1}$. For general loss, one can find $\mathbf{h}_\sigma$ using the IRLS method.

- Deletion
  - Find a coordinate (say k) among σ which violates the constraint maximally, and then update $\sigma = \sigma - \{k\}$ and update $\boldsymbol{\beta}$ by setting $\beta_k = 0$.
  - Recompute $\mathbf{h}_\sigma$.
  - Delete coordinates until $\boldsymbol{\beta} + \mathbf{h}_\sigma$ is feasible.

- Optimality check and addition
  - If $\boldsymbol{\beta}$ satisfies the optimality condition, the algorithm is terminated.
  - Otherwise, find the coefficient (say β_k) which violates the optimality condition most.
  - Update $\sigma = \sigma \cup \{k\}$ and then go to the optimization step.

- Remark
  - The Osborne's algorithm can fail to converge when p is larger than n.
  - In particular, when the cardinality of σ exceeds n, $(\mathbf{X}_{\sigma}^{'}\mathbf{X}_{\sigma})^{-1}$ does not exist.

Grafting (Perkins et al. 2003)

- It repeatedly modifies the set of nonzero coefficients (i.e. σ) as follows.
  - Start with $\sigma = \emptyset$.
  - Repeat until the solution is found.
    - Find a coefficient (say β_k) where the corresponding gradient has the largest absolute value.
    - Modify $\sigma = \sigma \cup \{k\}$.
    - Solve the LASSO problem with only coefficients in σ.
- The Grafting algorithm is computationally demanding since it solve the LASSO problem repeatedly.
- Also, similar to the Osborne's algorithm, Grafting can fail when $p > n$.

Stagewise forward selection: ϵ-boosting (Friedman 2001)

- Algorithm
  - Choose ϵ sufficiently small.
  - Select β_k whose gradient is the smallest.
  - Update $\boldsymbol{\beta} = \boldsymbol{\beta} + \epsilon \mathbf{e}_k$ where $\mathbf{e}_k$ is a p-dimensional vector where the k-th entry is 1 and the others are zero.
  - Keep iterating the above two steps until $\|\boldsymbol{\beta}\|_1$ exceeds λ.
- Efron et al. (2004) for square error loss (LARS) and Rossett et al. (2004) for general convex loss.
- It is simple and gives a solution path as a by-product.
- But for general convex loss except the square error loss, it may not converge to the solution.

4. **Coordinatewise gradient descent (CGD) algorithm**

- Recall that for LASSO we need to solve

$$\text{Minimize}_{\boldsymbol{\beta}} R(\boldsymbol{\beta})$$

  subject to $\|\boldsymbol{\beta}\|_1 \leq \lambda$.

- For simplicity, we let $\lambda = 1$. For other λ, we rescale the inputs by multiplying λ.

- Let $\mathbf{e}_k$ be the p-dimensional vector whose k-th entry is 1 and the others are zero (eg. $\mathbf{e}_1 = (1, 0, \ldots, 0)$) (called it an "coordinate vector").

- Let $\mathcal{E} = \{\mathbf{e}_k, -\mathbf{e}_k : k = 1, \ldots, p\}$.

- The LASSO problem can be restated as

$$\text{Minimize}_{\boldsymbol{\beta} \in co(\mathcal{E})} R(\boldsymbol{\beta})$$

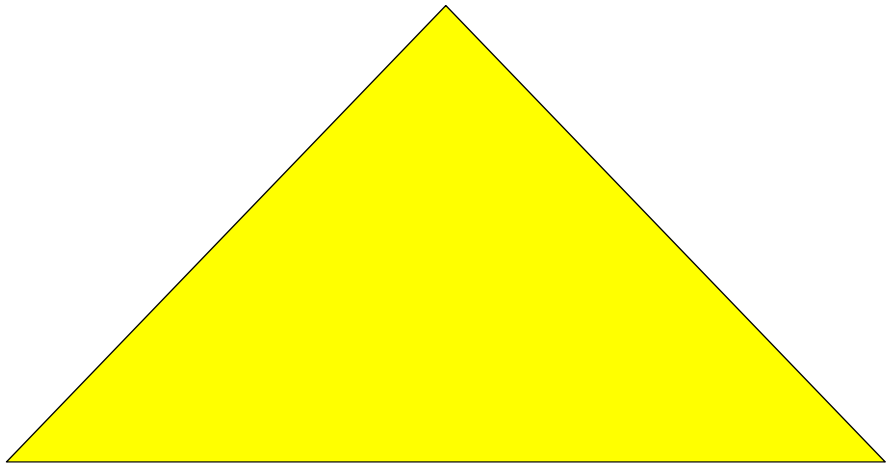
  where $co(\mathcal{E})$ is the convex hull of $\mathcal{E}$.

- The CGD algorithm updates β as follows.
  - Get the gradient vector $R^{(1)}(\boldsymbol{\beta}) = \partial R(\boldsymbol{\beta})/\partial \boldsymbol{\beta}$.
  - Choose the coordinate vector $\mathbf{e} \in \mathcal{E}$ which minimizes $\mathbf{e}' R^{(1)}(\boldsymbol{\beta})$.
  - Find the convex combination of $\boldsymbol{\beta}$ and $\mathbf{e}$ which minimizes R. That is, find $\alpha \in [0, 1]$ which minimizes $R(\alpha\boldsymbol{\beta} + (1 - \alpha)\mathbf{e})$.
  - Update $\boldsymbol{\beta} = \alpha\boldsymbol{\beta} + (1 - \alpha)\mathbf{e}$.

- That is
  - The LASSO problem is an optimization problem over the simplex of a given set of vertices.
  - The CGD algorithm repeatedly finds the optimal vertex (i.e smallest gradient) and takes the optimal convex combination (i.e. minimizing R).

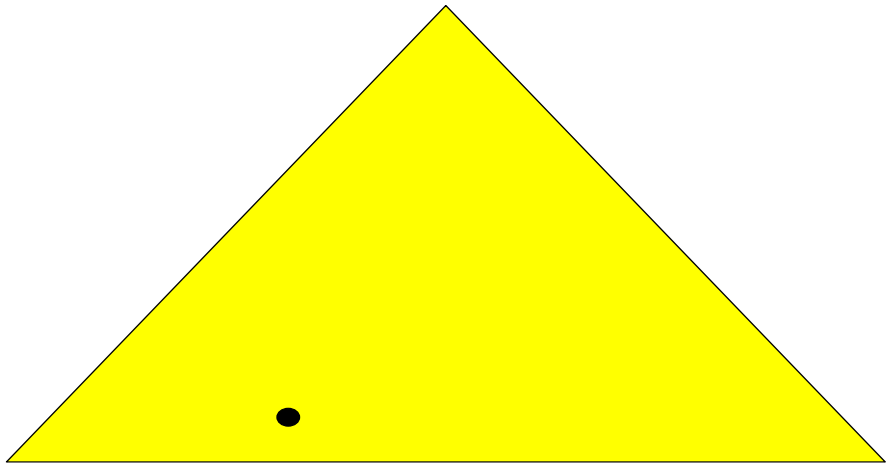
Illustration

- The number of vertices in $\mathcal{E}$ is 3.



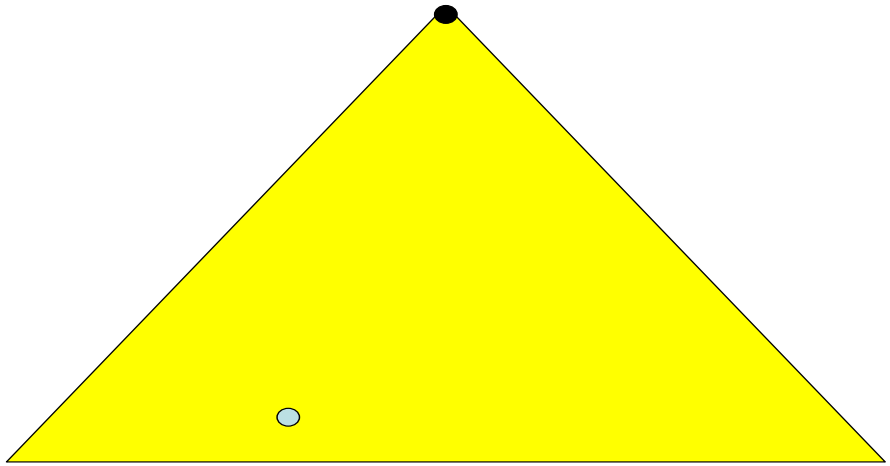
Illustration

- $\boldsymbol{\beta}$ is the current solution.



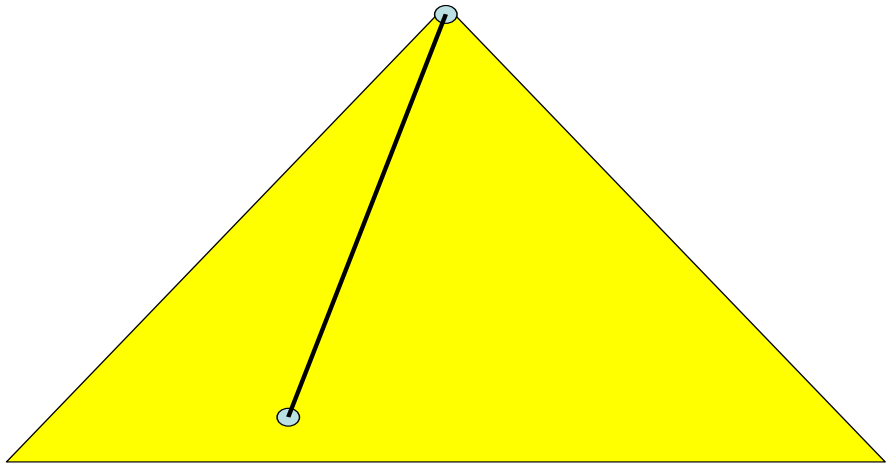
Illustration

- Find the vertex



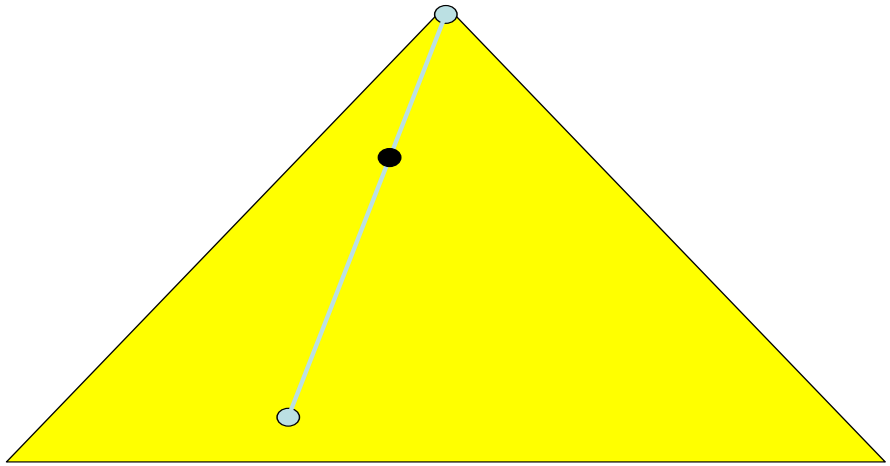
Illustration

- Take a convex combination



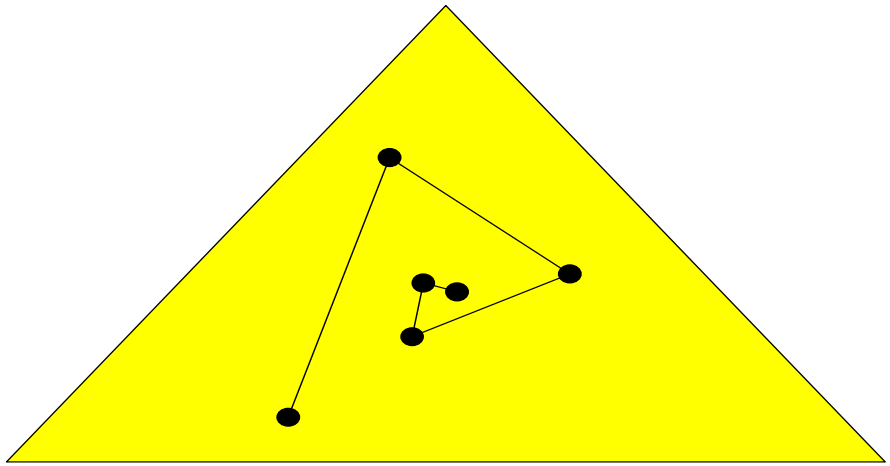
Illustration

- Move to the optimal convex combination.



Illustration

- A typical path



Remark

- In each update, we only need one-dimensional optimization for finding the optimal convex combination.
- No inversion of matrix is required.
- Hence, it can be used for very large dimensional data.

Convergence

- Let $\boldsymbol{\beta}_m$ be the solution obtained after the m-th iteration of the CGD algorithm.
- Let $\boldsymbol{\beta}^*$ be the optimal solution.
- Then

$$R(\boldsymbol{\beta}_m) - R(\boldsymbol{\beta}^*) = O(m^{-1}).$$

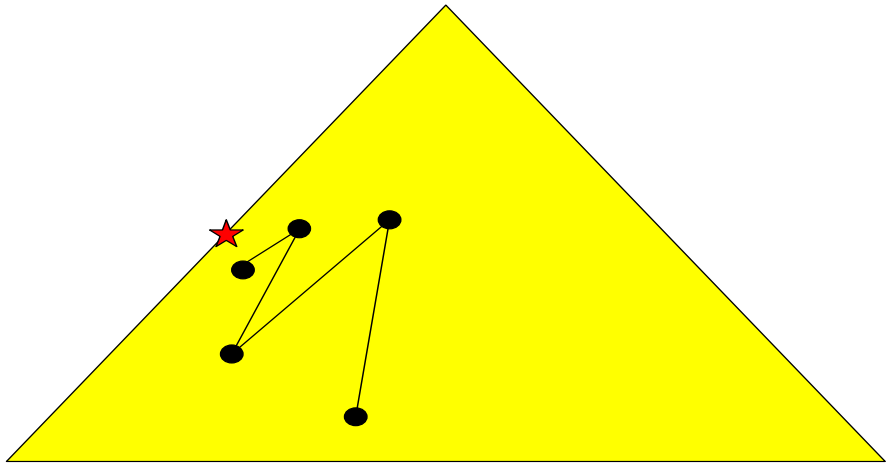
- Note: Apparently, $O(m^{-1})$ seems to be optimal for optimization over the convex set (Jones 1992, Barron 1993, Zhang 2003).

5. **Gradient LASSO algorithm**

A problem of the CGD algorithm

- For given $\boldsymbol{\beta}$, recall that $\sigma = \{j : \beta_j \neq 0\}$.
- We can say that the CGD algorithm consists of only the addition step.
- If $\beta_k \neq 0$ but $\beta_k^* = 0$, the CGD algorithm deletes β_k by repeatedly adding other βs.
- Hence, the CGD algorithm converges very slowly in this case.
- That is, when the optimal solution locates the boundary of the simplex of $\mathcal{E}$, the convergence is slow.

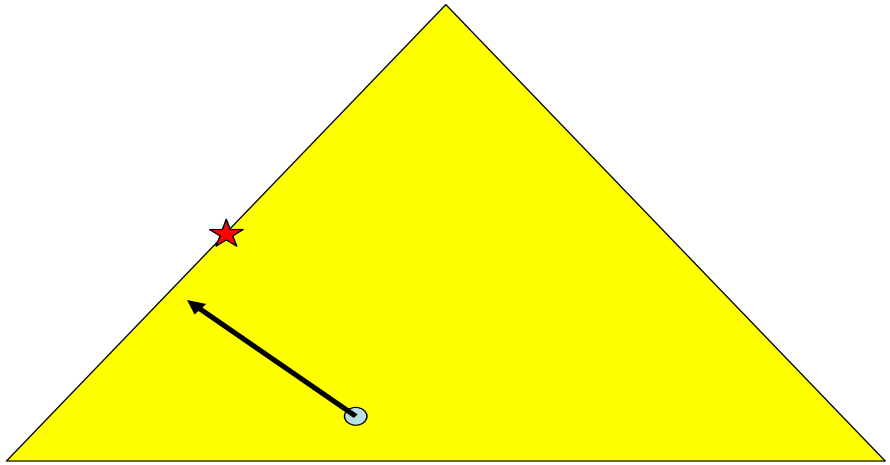
- Illustration



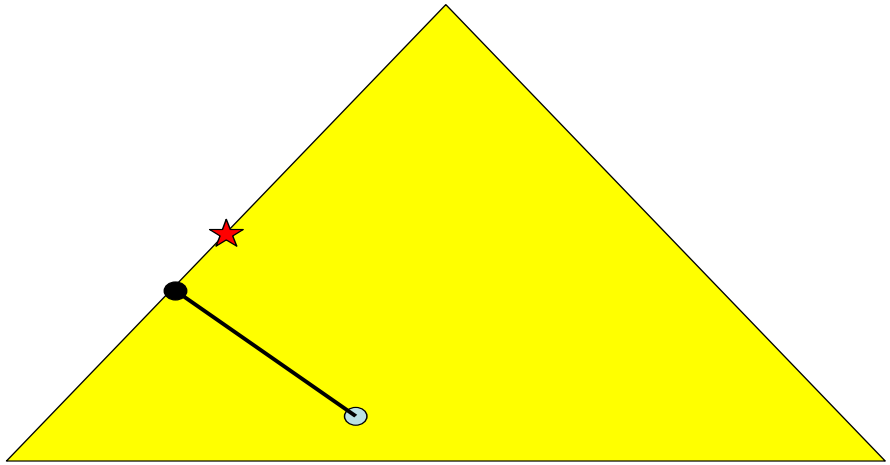
Speed up the CGD algorithm

- Instead of moving toward one of the vertices, move directly toward the optimal solution inside the simplex following the gradient direction as follows.

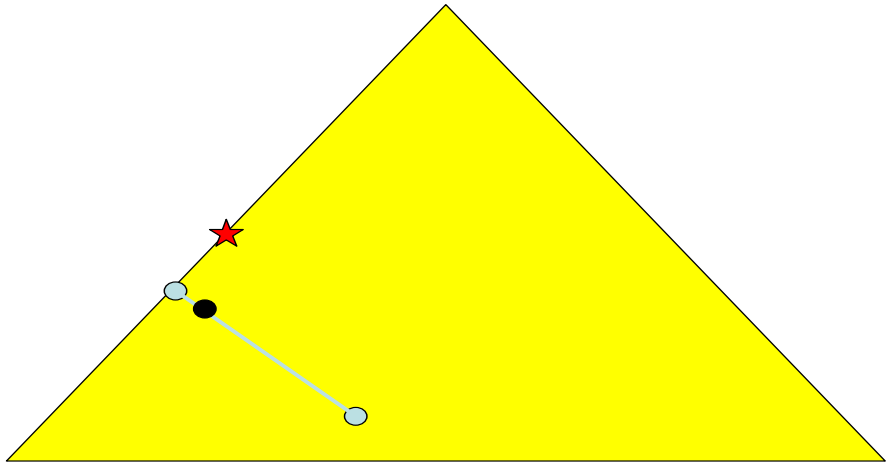
- Find the gradient direction



- Take a convex combination

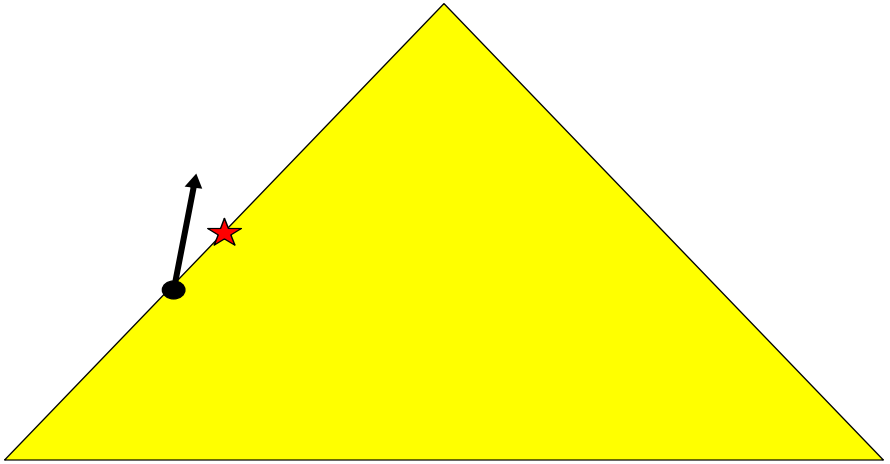


- Find the optimal convex combination

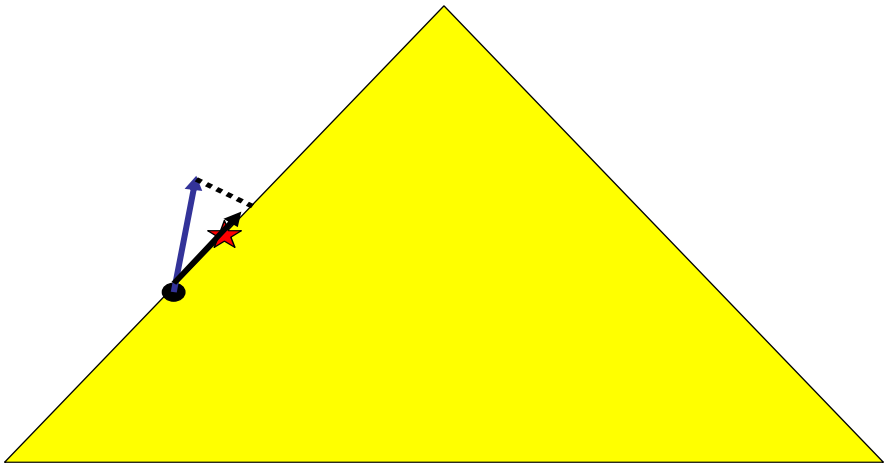


- In many cases, the gradient may direct the outside of the simplex.
- It happens when the current solution is on the boundary of the simplex.
- In this case, we project the gradient direction onto the simplex.

- Illustration
  - The gradient directs the outside of the simplex.



- Illustration
  - Project the gradient onto the simplex.



Gradient LASSO algorithm

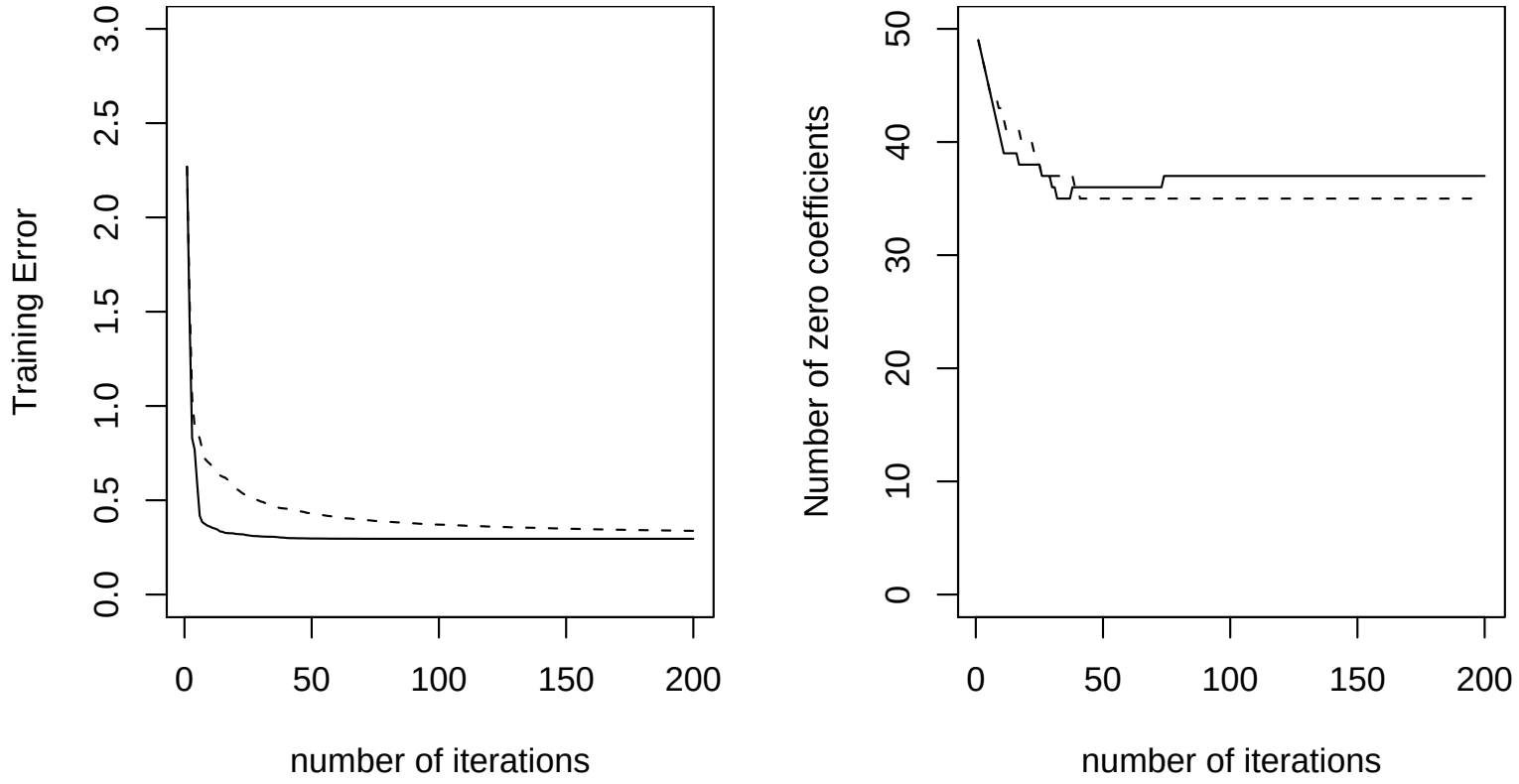
- Let $\mathcal{A} = \emptyset$ ($\mathcal{A}$: active set).
- The gradient LASSO algorithm consists of the two steps - addition and deletion.
  - Addition: Move the solution using the CGD algorithm, which may result in adding a new coordinate vector into $\mathcal{A}$.
  - Deletion: Move toward the (projected) gradient direction on the simplex of $\mathcal{A}$ (i.e. $co(\mathcal{A})$), which may result in deletion of a coordinate vector from $\mathcal{A}$.
- The gradient LASSO algorithm repeats the addition and deletion steps until the solution converges.
- Note: The gradient LASSO algorithm always converges faster that the CGD algorithm since the deletion step decreases the empirical risk.

Remark

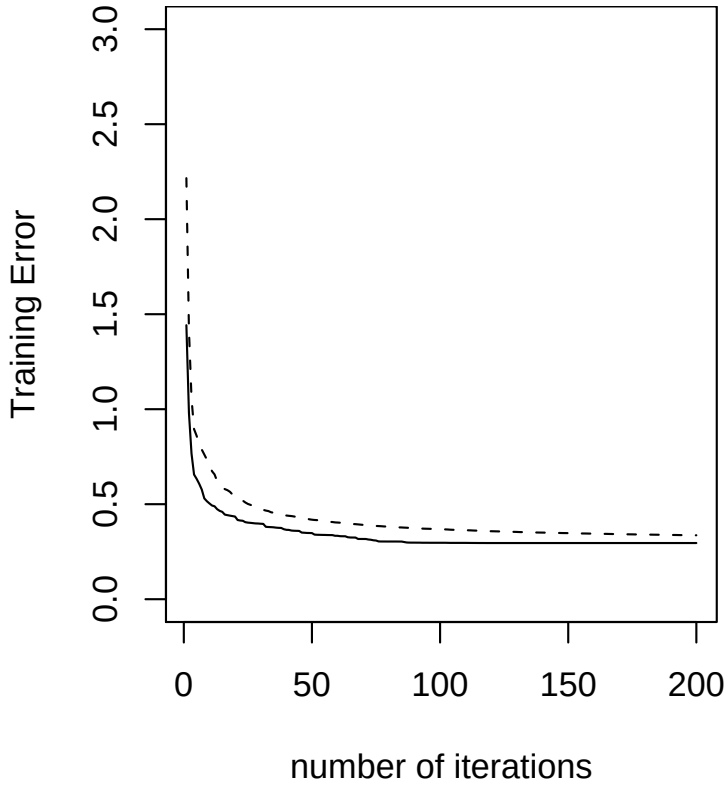
- The gradient LASSO algorithm does not require matrix inversion.
- In each iteration, we need two one-dimensional optimizations - one for addition and the other for deletion, which can be done easily.
- Hence, the gradient LASSO algorithm has all of the advantages of the CGD algorithm, and at the same time, it improves the speed of convergence significantly.

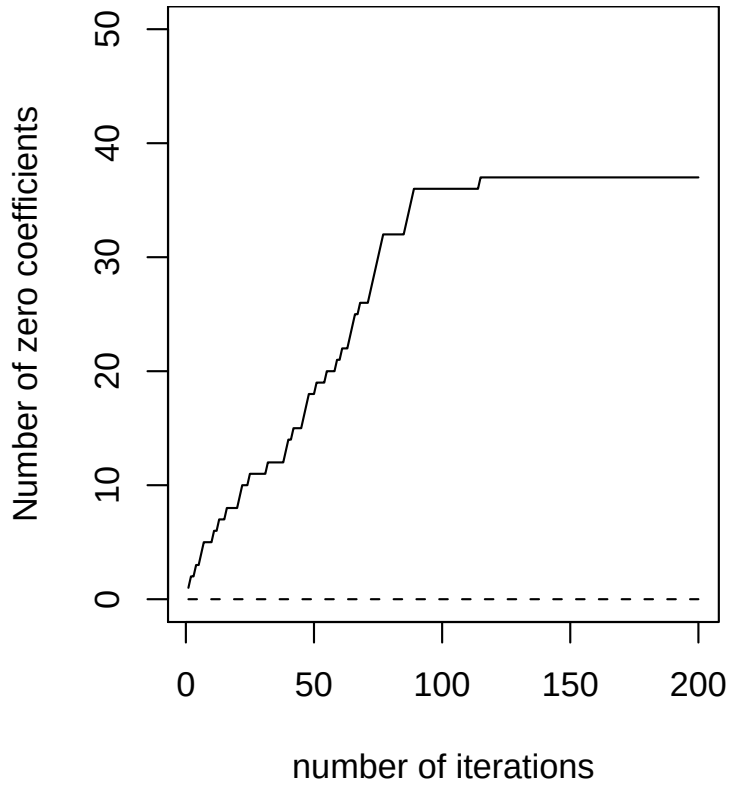
Comparison of the CGD and gradient LASSO algorithms

- Starting from the null model (i.e. $\beta_k = 0$ for all k).



- Starting from the full model (i.e. $\beta_k = 3/50$).





- The CGD and gradient LASSO give similar results when we start from the null model.
- However, the CGD algorithm completely fails to delete unnecessary coefficients when we start from the full model.
- This is because the CGD algorithm keeps adding coefficients.
- It is interesting to see that even if the number of zero coefficients are quite different in the full model case, the empirical risks are very close.
- This observation suggests that finding an approximated solution (one which minimizes the empirical risk approximately) is not enough for sparse learning, in particular for variable selection.

6. **Simulation: Gradient LASSO vs Osborne**

- Logistic regression
- Sample size is 20
- Only three coefficients are set to be nonzero.
- 100 repetition

- Results

p	λ	Method	# NA	Time	Train error	# zero
50	1	Osborne	0	0.04	0.4240	45.38
		Gradient	0	0.13	0.4240	45.38
	10	Osborne	29	0.04	0.0242	37.63
		Gradient	0	0.48	0.0242	37.49
100	1	Osborne	0	0.06	0.4019	95.03
		Gradient	0	0.18	0.4019	95.02
	10	Osborne	63	0.07	0.0107	87.35
		Gradient	0	0.48	0.0108	86.92

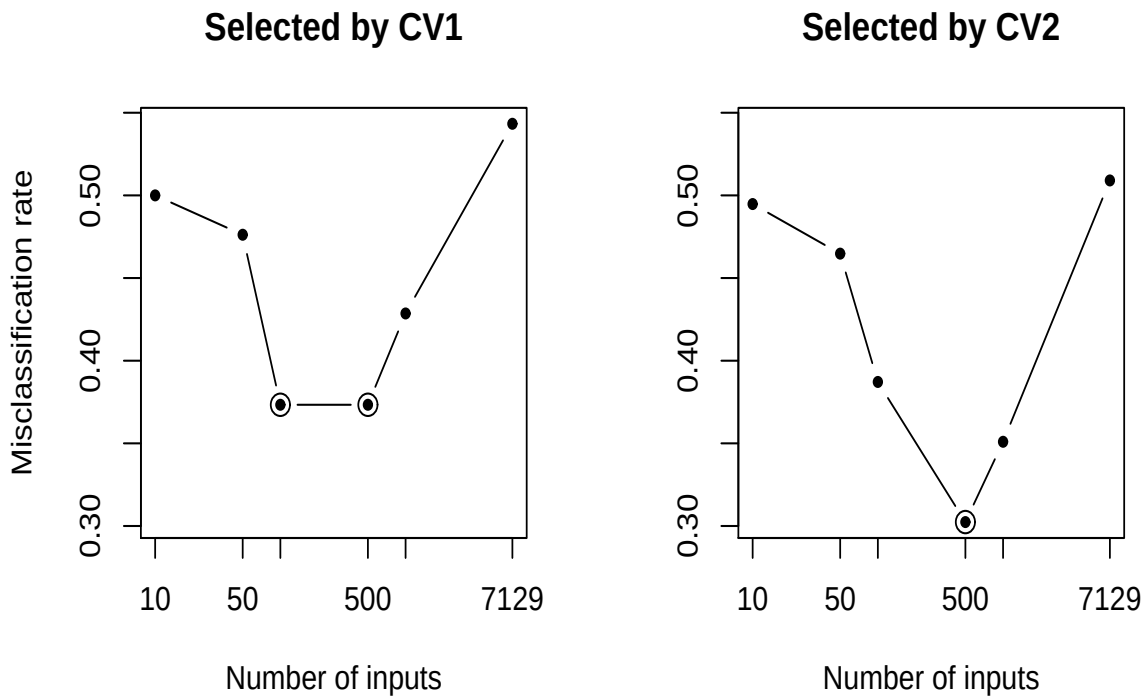
- The Osborne's algorithm is faster.
- But, it fails quite often when λ is large.
- In contrast, the gradient LASSO algorithm never fails and gives similar results as those of Osborne's even though it needs more computing time.

7. **Analysis of gene expression data**

- We analyzed the NCI60 data set, which consists of
  - $n = 60$
  - $p = 7129$ (number of genes)
  - # of class $= 7$
- The multiclass logistic regression model is used.
- The misclassification error is measured by repeating random split of the data to training (70%) and test (30%) data sets 100 times.
- The regularization parameter λ is selected using the five-fold cross validation with 0-1 loss (CV1) and logistic loss (CV2).

- Objective of the analysis
  - A popular approach for gene expression data is to select a small number of genes a priori using a prescreening measure such as the F-ratio, and then construct a predictive model.
  - This saves computing time and makes it possible to use various advanced learning algorithms.
  - But, it is not clear how the prescreening affects the accuracy.
  - An objective of this analysis is to see the effect of prescreening to prediction accuracy.

- Results


Selected by CV1
Misclassification rate
0.30
0.40
0.50
10
50
500
7129
Number of inputs
Selected by CV2
0.30
0.40
0.50
10
50
500
7129
Number of inputs

- Remark
  - The optimal number of genes is 500, which is still large.
  - Our results suggest that deleting too many genes a priori can degrade the accuracy significantly.
  - So, we recommend using prescreening procedures not for selecting important genes but for deleting noisy genes.
  - Hence, computing algorithms which can process large dimensional data are still necessary for better accuracy even after a prescreening process.
  - The gradient LASSO algorithm is one of such algorithms.

8. **Concluding Remarks**

- There are various sparse learning methods.
- Examples are fused LASSO (Tibshirani et al. 2005), grouped LASSO (Yuan and Lin, 2004), blockwise sparse regression (Kim et al. 2006), SCAD (Fan and Li, 2001) and elastic net (Zou and Hastie, 2004).
- We have seen that for sparse learning methods, crude approximated solutions may be significantly misleading, especially for variable selection.
- Hence, it is worth pursuing to develop efficient and globally convergent computational algorithms for these methods, particularly for large dimensional data.

The R-library of the gradient LASSO algorithm can be downloaded at "http://idea.snu.ac.kr/Research/glassojskim/glasso.htm".