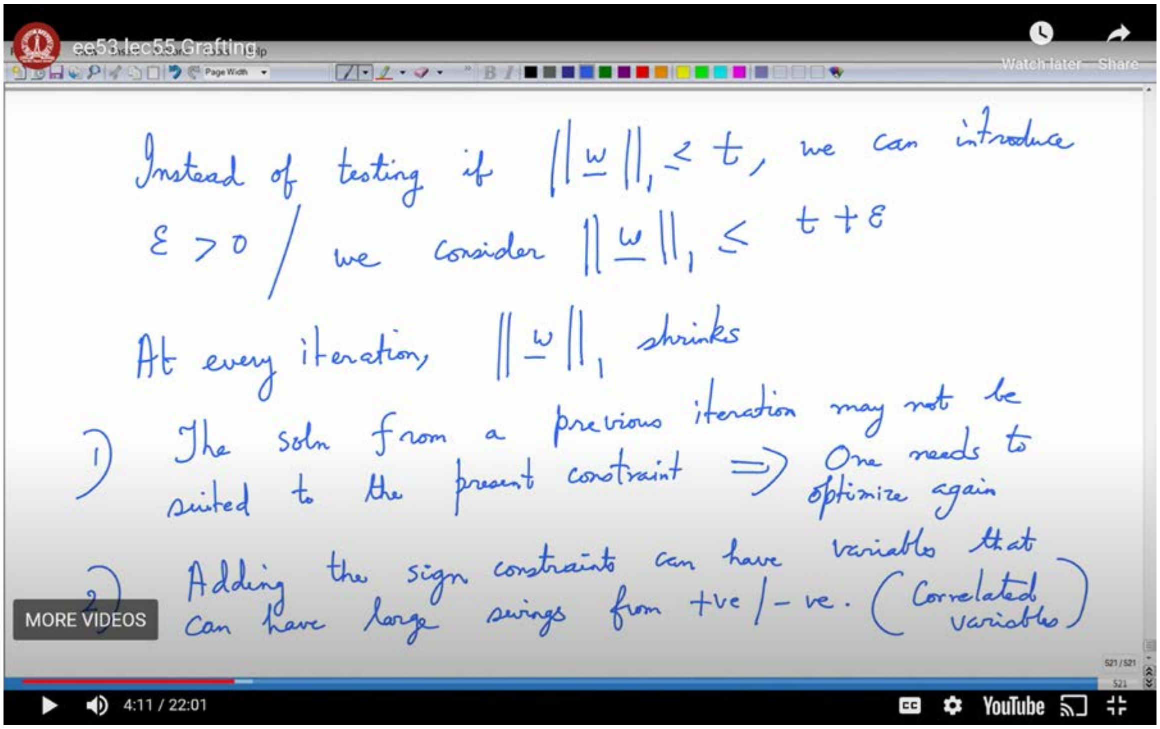
Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani
Department of Electronic System Engineering
Indian Institute of Science, Bengaluru

Lecture – 55
Grafting

Let's delve deeper into the Lasso procedure, incorporating some practical heuristics to refine our approach. Instead of rigidly constraining the L_1 norm to be less than or equal to a fixed value t, we can modify our constraint to allow the L_1 norm to be less than or equal to t + ϵ, where ϵ is a small positive quantity. This adjustment can potentially accelerate our convergence toward the optimal solution.

(Refer Slide Time: 04:11)



Instead of testing if $\|\underline{w}\|_1 \leq t$, we can introduce $\varepsilon > 0$ / we consider $\|\underline{w}\|_1 \leq t + \varepsilon$

At every iteration, $\|\underline{w}\|_1$ shrinks

1) The soln from a previous iteration may not be suited to the present constraint $\Rightarrow$ One needs to optimize again

2) Adding the sign constraints can have variables that can have large swings from +ve / − ve. (Correlated variables)

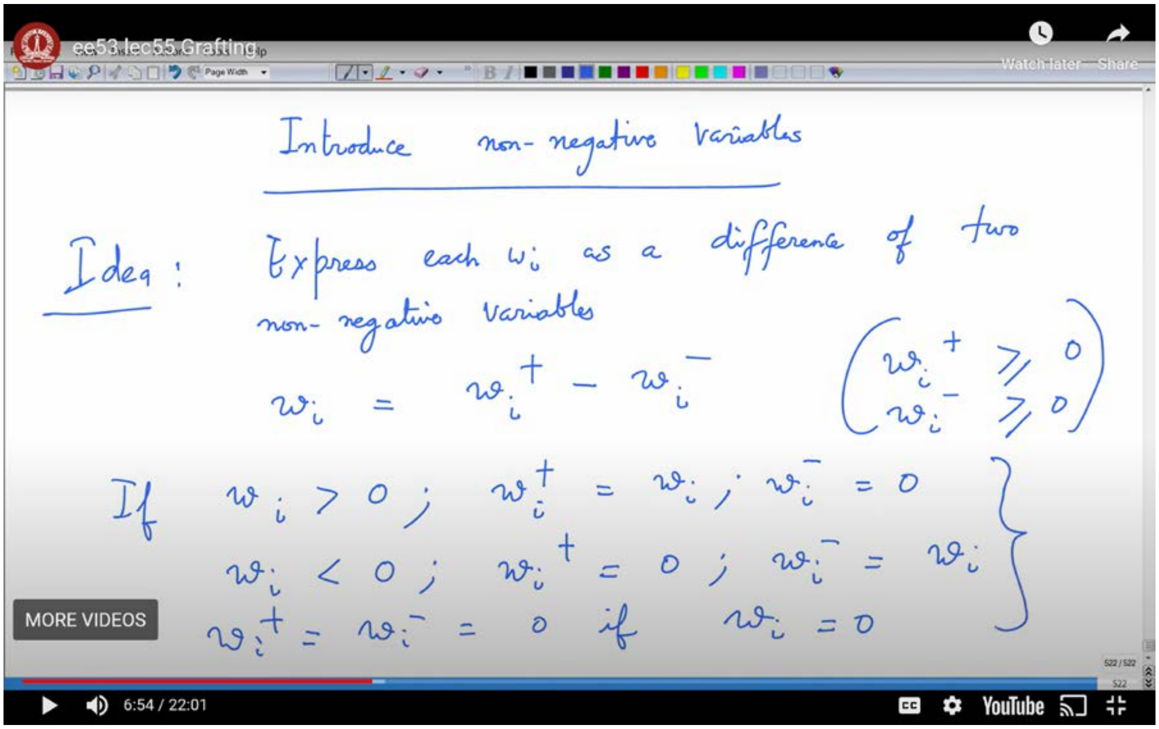
To formalize this, instead of strictly enforcing $|w|_1 \leq t$, we consider $|w|_1 \leq t + \epsilon$, where ϵ is a positive constant. By incorporating this relaxed constraint, we hope to facilitate a faster approach to the solution.

There are two critical points to consider with this approach:

1. Constraint Adaptation: The solution from a previous iteration, which was obtained with a specific set of constraints, may not be suitable for the current iteration's constraints. Consequently, this necessitates re-optimization, as the initial conditions from the previous step may not provide a good starting point for the current step.

2. Variable Swings: Adding sign constraints can lead to significant fluctuations in the variables, with values swinging from highly positive to highly negative. This issue is especially pronounced when dealing with correlated variables, adding complexity to the optimization process.

(Refer Slide Time: 06:54)



Introduce non-negative variables

Idea : Express each w_i as a difference of two non-negative variables

$$w_i = w_i^+ - w_i^- \qquad \begin{pmatrix} w_i^+ \geq 0 \\ w_i^- \geq 0 \end{pmatrix}$$

$$\left.\begin{array}{l} \text{If } \; w_i > 0 ; \; w_i^+ = w_i ; \; w_i^- = 0 \\ w_i < 0 ; \; w_i^+ = 0 ; \; w_i^- = w_i \\ w_i^+ = w_i^- = 0 \; \text{if} \; w_i = 0 \end{array}\right\}$$

To address these challenges, we can introduce non-negative variables. Since the weights can be both positive and negative, we can represent each weight as the difference between two non-negative variables. This technique effectively doubles the number of variables but simplifies the linear constraints.

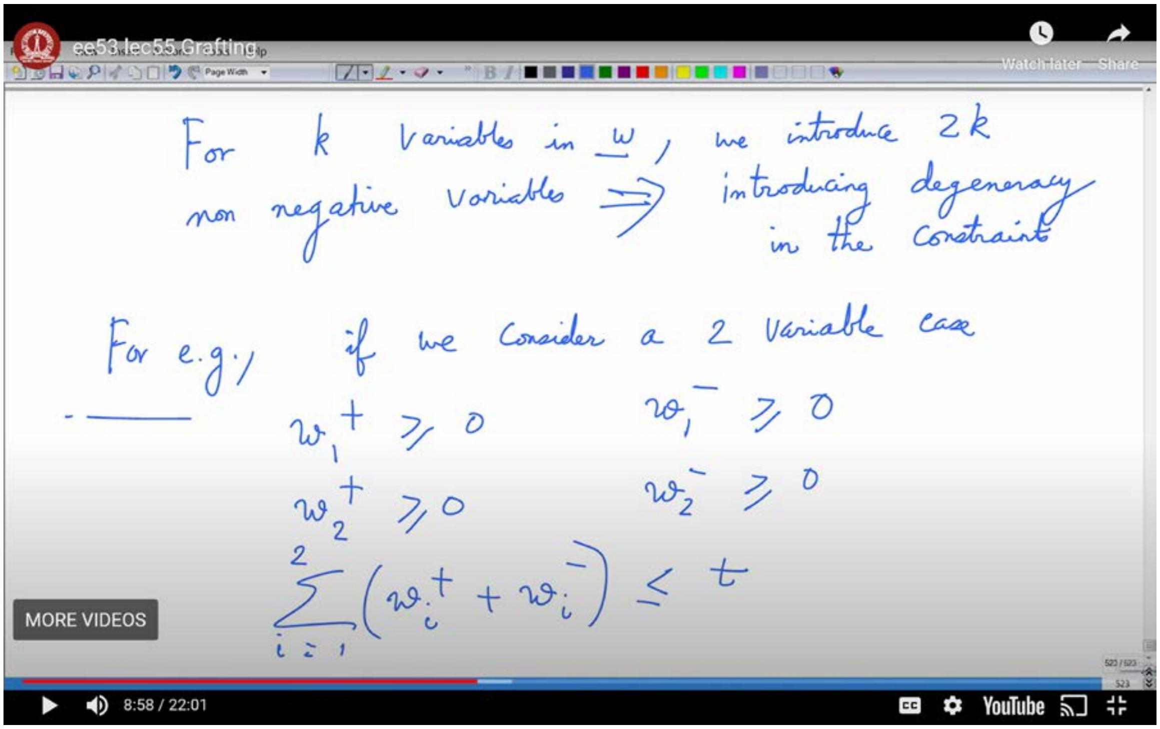
Here's how it works:

Representation of Weights: Each weight w_i is expressed as the difference between two non-negative variables w_i^+ and w_i^-. Specifically:

$$w_i = w_i^+ - w_i^-$$

where $w_i^+ \geq 0$ and $w_i^- \geq 0$.

(Refer Slide Time: 08:58)



For k variables in $\underline{w}$, we introduce $2k$ non negative variables $\Rightarrow$ introducing degeneracy in the constraints

For e.g., if we consider a 2 variable case

$$w_1^+ \geq 0 \qquad w_1^- \geq 0$$

$$w_2^+ \geq 0 \qquad w_2^- \geq 0$$

$$\sum_{i=1}^{2} (w_i^+ + w_i^-) \leq t$$

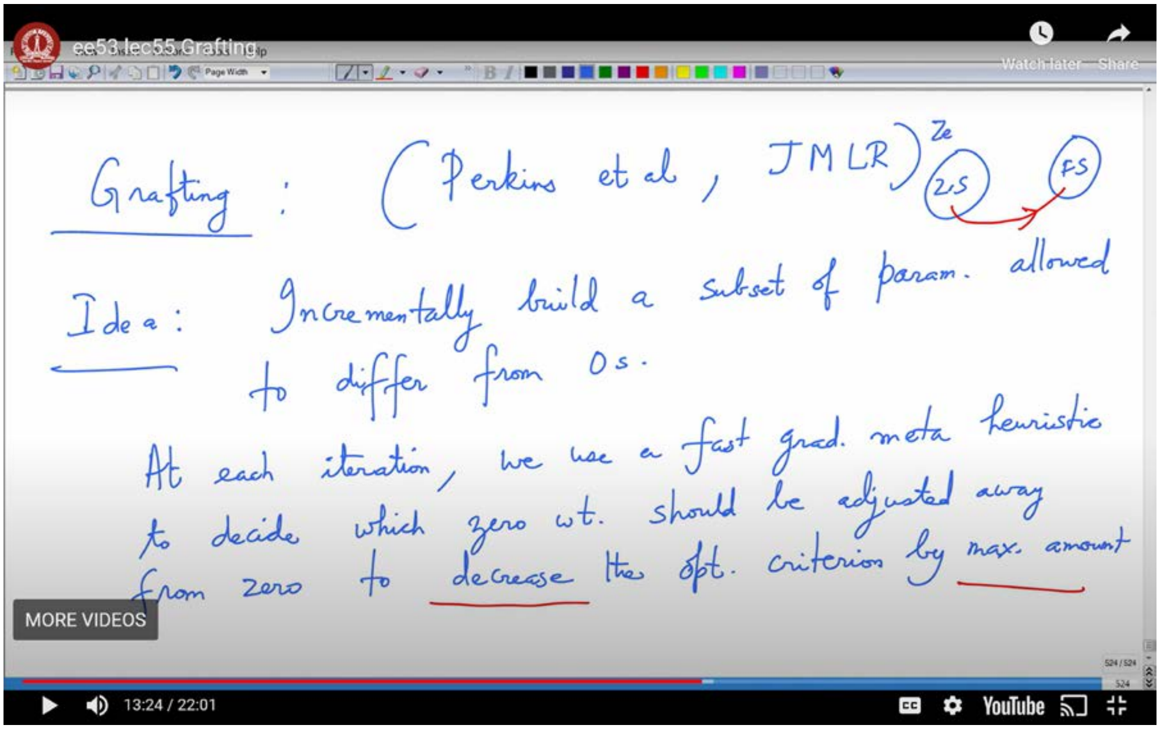
Cases:

- If w_i is positive, set $w_i^+ = w_i$ and $w_i^- = 0$.
- If w_i is negative, set $w_i^+ = 0$ and $w_i^- = -w_i$.
- If w_i is zero, set both w_i^+ and w_i^- to 0.

This method of expressing weights as differences of non-negative variables helps in managing the constraints more effectively and simplifies the optimization process, despite the increase in the number of variables.

Let's explore the implications of introducing additional variables into our optimization problem. When dealing with k variables in the weight vector w, if we introduce 2k non-negative variables, we essentially add a layer of degeneracy to the constraints. This means we create a scenario where constraints become less restrictive or "degenerate."

(Refer Slide Time: 13:24)



Grafting : (Perkins et al, JMLR) Ze ZS → FS

Idea : Incrementally build a subset of param. allowed to differ from 0s.

At each iteration, we use a fast grad. meta heuristic to decide which zero wt. should be adjusted away from zero to decrease the opt. criterion by max. amount

To illustrate this, consider a scenario with two variables. We would introduce four non-negative variables: $w_1^+ \geq 0$, $w_1^- \geq 0$, $w_2^+ \geq 0$, and $w_2^- \geq 0$. The constraint can then be expressed as:

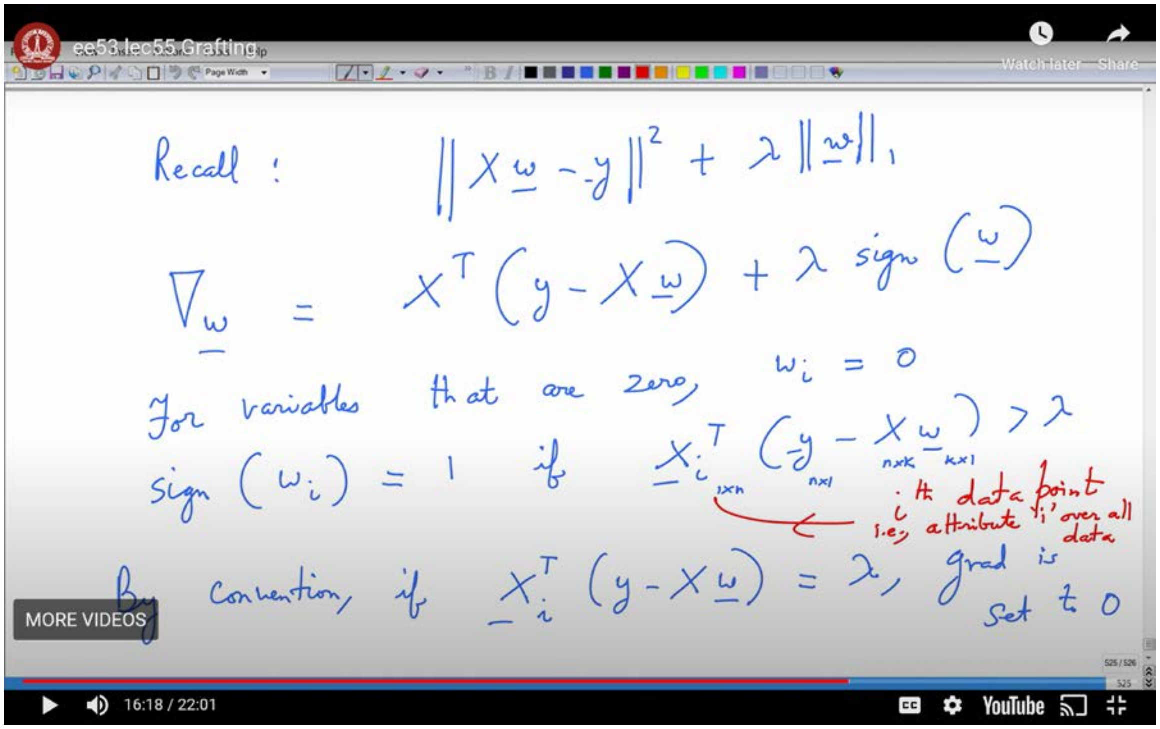
$$w_1^+ + w_1^- + w_2^+ + w_2^- \leq t$$

This represents a linear constraint and is considered degenerate because it is much less restrictive than the original set of constraints.

While these techniques simplify the problem, they come at the cost of introducing additional variables, which must be factored into the optimization process.

One notable technique in this context is grafting, which focuses on solving the unconstrained problem formulation. The goal here is to decide which weights to adjust from the zero set to maximize the reduction in the optimization criterion. The zero set refers to variables currently set to zero, while the free set comprises variables that can be adjusted. The concept of grafting involves iteratively moving variables from the zero set to the free set, optimizing the objective function as much as possible at each step.

(Refer Slide Time: 16:18)



Recall : $\|X\underline{w} - y\|^2 + \lambda\|\underline{w}\|_1$

$\nabla_{\underline{w}} = X^T(y - X\underline{w}) + \lambda \text{ sign}(\underline{w})$

For variables that are zero, $w_i = 0$

$\text{sign}(w_i) = 1$ if $\underset{1\times n}{\underline{X}_i^T}(\underset{n\times 1}{\underline{y}} - \underset{n\times k}{X}\underset{k\times 1}{\underline{w}}) > \lambda$

i th data point
i.e., i th attribute 'i' over all data

By convention, if $\underline{X}_i^T(y - X\underline{w}) = \lambda$, grad is set to 0

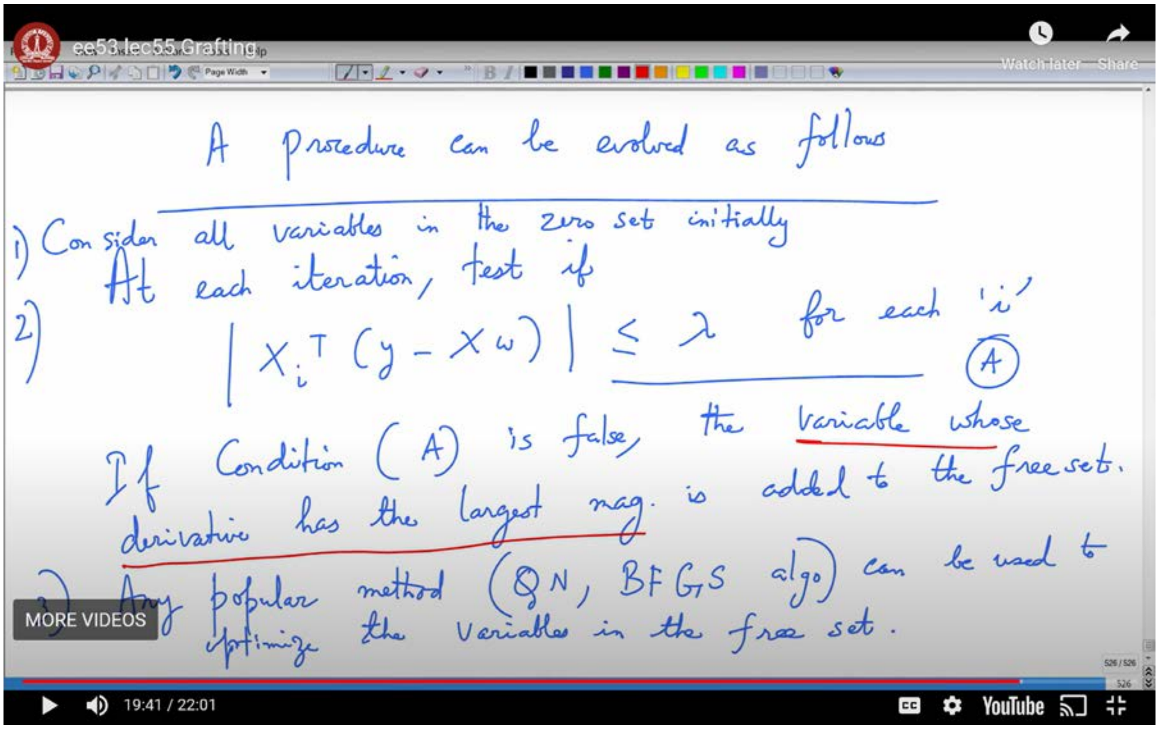
This approach, known as grafting, was originally developed by Perkins and colleagues and published in the Journal of Machine Learning Research in 2003. The basic idea is to incrementally build a subset of parameters that can differ from zero. This is done using a metaheuristic approach, specifically a fast gradient-based heuristic. At each iteration, the heuristic determines which zero weight should be adjusted to achieve the maximum reduction in the optimization criterion.

To formalize this, recall the constraint:

$$|xw - y|_2^2$$

subject to L1 constraints on the weights.

(Refer Slide Time: 19:41)



A procedure can be evolved as follows

1) Consider all variables in the zero set initially

2) At each iteration, test if

$$\left| x_i^T (y - Xw) \right| \le \lambda \quad \text{for each 'i'} \quad \text{Ⓐ}$$

If Condition (A) is false, the variable whose derivative has the largest mag. is added to the free set.

3) Any popular method (QN, BFGS algo) can be used to optimize the variables in the free set.

Taking the gradient with respect to the weights using straightforward matrix calculus, we get the following: For weights that are zero (wi = 0), the sign of wi is determined by whether $x_i^T(y - xw)$ is greater than λ. Specifically:

$$\text{sign}(w_i) = \begin{cases} 1 & \text{if } \mathbf{x}_i^T(\mathbf{y} - \mathbf{xw}) > \lambda \\ -1 & \text{if } \mathbf{x}_i^T(\mathbf{y} - \mathbf{xw}) < -\lambda \\ 0 & \text{if } \mathbf{x}_i^T(\mathbf{y} - \mathbf{xw}) = \lambda \end{cases}$$

Here, x_i refers to the i-th data point, and λ is a regularization parameter.

By convention, if $x_i^T(y - xw) = \lambda$, the gradient is set to zero. This adjustment process is crucial for moving variables from the zero set to the free set, and ensuring that each iteration meets the conditions for optimality.

If Condition A is not satisfied, we then select the variable with the largest magnitude of its derivative and move it to the free set. It's important to emphasize that initially, all variables are considered to be in the zero set. At each iteration, we check whether Condition A is met. If it is not, we add the variable with the largest derivative magnitude to the free set. Once in the free set, we can optimize these variables using any popular method, such as the quasi-Newton method or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.

During each quasi-Newton iteration, we recompute the free set. The process continues until we arrive at a set of non-zero variables and zero-valued variables with the largest magnitude of derivatives. Essentially, we examine the derivatives, identify the one with the largest magnitude, and move that variable to the free set. This is the essence of the grafting method, which, in practice, proves to be quite effective.

Here’s a summary of the procedure: Begin with all variables in the zero set. At each iteration, check if Condition A is satisfied. If not, move the variable with the largest derivative magnitude to the free set. Then, perform optimization on this updated free set using techniques such as quasi-Newton methods. This process is repeated, transferring variables from the zero set to the free set, until no further variables can be moved.

One can explore the convergence properties and stability of these algorithms, but these are topics for an optimization course. My goal here was to provide a foundational understanding of L_1 regularization, the constraints involved, and basic optimization setups and algorithms related to L_1 constraints. This foundational knowledge should help in appreciating the role of sparsity constraints in real-world problems.

A detailed exploration of L_1 regularization theory could indeed be a subject for a future course, given the breadth of techniques available. For now, we have covered the basics of L_1 regularization, and I hope these techniques will be useful in your practical applications.