OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Efficient Designs Of SLOPE Penalty Sequences In Finite Dimension

Yiliang Zhang ZYLTHU14@SAS.UPENN.EDU
Zhiqi Bu ZBU@SAS.UPENN.EDU
Philadelphia, USA

Abstract

In linear regression, SLOPE is a new convex analysis method that generalizes the Lasso via the sorted ℓ_1 penalty: larger fitted coefficients are penalized more heavily. This magnitude-dependent regularization requires an input of penalty sequence $\boldsymbol{\lambda}$, instead of a scalar penalty as in the Lasso case, thus making the design extremely expensive in computation. In this paper, we propose two efficient algorithms to design the possibly high-dimensional SLOPE penalty, in order to minimize the mean squared error. For Gaussian data matrices, we propose a first order Projected Gradient Descent (PGD) under the Approximate Message Passing regime. For general data matrices, we present a zero-th order Coordinate Descent (CD) to design a sub-class of SLOPE, referred to as the k-level SLOPE. Our CD allows an useful trade-off between the accuracy and the computation speed. We demonstrate the performance of SLOPE with our designs via various experiments on synthetic data and real-world datasets.

1. Introduction

In linear regression, we aim to find an accurate estimator $\hat{\boldsymbol{\beta}}$ of the unknown truth $\boldsymbol{\beta}$ from

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{w}.$$

where the response $\boldsymbol{y} \in \mathbb{R}^n$, the data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, the true parameter $\boldsymbol{\beta} \in \mathbb{R}^p$ and the noise $\boldsymbol{w} \in \mathbb{R}^n$. Specifically, in high dimension where $p > n$, ordinary linear regression fails to find a unique solution and ℓ_1-related regularization is usually introduced to find sparse estimators, including the Lasso [19], elastic net [22], (sparse) group Lasso [20], adaptive Lasso [21] and the recent SLOPE [4]:

$$\widehat{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{b}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|_2^2 + \sum_{i=1}^{p} \lambda_i |b|_{(i)} \tag{1.1}$$

Here $\sum_{i=1}^p \lambda_i |b|_{(i)}$ is the *sorted* ℓ_1 *norm* of $\boldsymbol{b}$ governed by the penalty vector $\boldsymbol{\lambda} \in \mathbb{R}^p$ with $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$, and $|b|_{(i)}$ is the ordered statistics of absolute values $|b_i|$ such that $|b|_{(1)} \geq \cdots \geq |b|_{(p)}$.[1] The sorting step in the norm allows SLOPE to work similarly to the taxation: assigning larger thresholds to larger fitted coefficients.

Many desirable properties have been proven for SLOPE. For example, SLOPE is a convex optimization that can be solved by existing subgradient descent and the proximal gradient descent; SLOPE controls the false discovery rate in the case of independent predictors. However, questions such as what posterior distribution does SLOPE solution follow, can we characterize statistics (e.g. the false discovery rate and true positive rate) exactly, whether SLOPE has better estimation error than the Lasso and how to design the $\mathbb{R}^p$ SLOPE penalty are not answered until recently [3, 5, 10].

1. The Lasso is a sub-case of SLOPE when $\lambda_1 = \cdots = \lambda_p$.

© Y. Zhang & Z. Bu.

The substantial difficulty imposed by sorted penalty impedes the general application of SLOPE by two means. From the practical point of view, tuning a $\mathbb{R}^p$ penalty can be extremely costly for large p and naive methods that work for the Lasso, such as the grid search, renders not pragmatic. From a theoretical perspective, the sorted norm is complicated as the effect of thresholding of SLOPE is, unlike the Lasso, *non-separable* and *data-dependent*, making the analysis much involved.

We give a computationally efficient framework to design the SLOPE penalty sequence $\boldsymbol{\lambda} \in \mathbb{R}^p$ which corresponds to an estimator $\hat{\boldsymbol{\beta}}(\boldsymbol{\lambda})$ that minimizes the estimation error. To be more specific, we derive the gradient of penalty for SLOPE under the Approximate Message Passing (AMP) regime [1, 2, 8, 9] and propose to use projected gradient descent to search for optimal penalty sequences. For general data matrcies, we propose the *k-level* SLOPE. In words, k-level SLOPE is a sub-class of SLOPE, where the p elements in $\{\lambda_i\}$ have only k unique values. Under this definition, the general SLOPE is p-level SLOPE and the Lasso is indeed 1-level SLOPE. Additionally, k-level SLOPE is a sub-class of $(k+1)$-level SLOPE, and larger k leads to better performance but requires longer computation time. We illustrate that even 2-level SLOPE ($k=2$) can outperform the Lasso. [2]

1.1. Notations

We start by introducing the proximal operator of SLOPE,

$$\mathrm{prox}_{J_{\boldsymbol{\theta}}}(\boldsymbol{y}) := \underset{\boldsymbol{b}}{\mathrm{argmin}} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{b}\|^2 + J_{\boldsymbol{\theta}}(\boldsymbol{b}), \tag{1.2}$$

where $J_{\boldsymbol{\theta}}(\boldsymbol{b}) := \sum_{i=1}^{p} \lambda_i |b|_{(i)}$ and the proximal operator indeed solves (1.1) with an identity data matrix. This operator is the building block that is iteratively applied to derive the SLOPE estimator. We note that there is no closed form of $\mathrm{prox}_{J_{\boldsymbol{\theta}}}(\boldsymbol{x})$ but it can be efficiently computed as in [4, Algorithm 3]. Next we denote the mean squared error ($\mathtt{MSE}$) as $\mathtt{MSE}(\boldsymbol{u}, \boldsymbol{v}) := \|u - v\|^2$. Two performance measures that are investigated in this work are the prediction error, $\mathtt{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$, and the estimation error, $\mathtt{MSE}(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$.

2. SLOPE penalty design under AMP regime

2.1. Computing gradients with respect to penalty

We introduce a special regime of the Approximate Message Passing (AMP) [1, 2, 8, 9] for SLOPE [6], within which the SLOPE estimator can be asymptotically exactly characterized. A similar regime is the case when Convex Gaussian Min-max Theorem (CGMT) [7, 16–18] applies, which shares similar assumption as those in AMP. We then derive the gradient of $\mathtt{MSE}(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$ with respect to the penalty $\boldsymbol{\lambda}$ and optimize our penalty design iteratively. Generally speaking, AMP is a class of gradient-based optimization algorithms that mainly work on independent Gaussian random data matrix, offering both a sequence of estimators that converge to the true minimizer and a distributional characterization of the latter.

We assume the noise $\boldsymbol{w}$ has zero mean and variance σ_w^2. More assumptions of SLOPE AMP can be found in appendix A. We base our analysis on two key quantities $(\boldsymbol{\alpha}, \tau)$ which are the unique solution to the following equations, namely the calibration and the state evolution in the AMP regime (see [5, 10]):

2. The general SLOPE is p-level SLOPE and the Lasso is indeed 1-level SLOPE.

$$\boldsymbol{\lambda} = \boldsymbol{\alpha}\tau \left(1 - \frac{1}{n}\mathbb{E}\left\|\mathrm{prox}\left(\boldsymbol{\beta} + \tau \boldsymbol{Z}; \boldsymbol{\alpha}\tau\right)\right\|_0^*\right) \tag{2.1}$$

$$\tau^2 = \sigma_w^2 + \frac{1}{\delta p}\mathbb{E}\left\|\mathrm{prox}\left(\boldsymbol{\beta} + \tau \boldsymbol{Z}; \boldsymbol{\alpha}\tau\right) - \boldsymbol{\beta}\right\|^2 \tag{2.2}$$

Here $\|\cdot\|_0^*$ is a modified ℓ_0 norm that counts the unique non-zero absolute values in a vector and $\boldsymbol{Z} \in \mathbb{R}^p$ follows i.i.d. standard normal. We denote $\delta := \lim_p n/p$ as the aspect ratio or sampling ratio and $\epsilon := \mathrm{plim}\,|\{j : \beta_j \neq 0\}|/p$ where plim is the probability limit. We now introduce some properties that are useful to derive the desirable $\boldsymbol{\lambda}$. By [5, Proposition 2.3], the calibration (2.1) describes a bijective, monotone and parallel mapping between $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ [5, Proposition 2.3], which allows us to work with $\boldsymbol{\alpha}$ easily instead of $\boldsymbol{\lambda}$. By [5, Theorem 1], the state evolution (2.2) can be solved via a fixed point recursion, which converges to the unique solution $\tau(\boldsymbol{\alpha})$ monotonically under any initial condition.

To demonstrate our design of $\boldsymbol{\lambda} \in \mathbb{R}^p$ in SLOPE, we start with quoting [6, Corollary 3.4]:

$$\mathrm{plim}\,\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|^2/p = \delta(\tau^2 - \sigma_w^2). \tag{2.3}$$

Hence minimizing $\mathtt{MSE}(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$ is equivalent to minimizing τ, which depends on $\boldsymbol{\alpha}$ and leads to differentiating (2.2) against each of α_i for $i \in [p]$. Next, we use the gradient information to descend (with the projection elaborated in Algorithm 3) till convergence. Once the minimizing $\boldsymbol{\alpha}$ is obtained, we leverage the calibration (2.1) to find the corresponding $\boldsymbol{\lambda}(\boldsymbol{\alpha})$.

We shorthand $\mathrm{prox}_{J_{\boldsymbol{b}}}(\boldsymbol{a})$ by using $\boldsymbol{\eta}(\boldsymbol{a}; \boldsymbol{b})$. In particular $\mathrm{prox}_{J_{\boldsymbol{\alpha}\tau}}(\boldsymbol{\beta} + \tau\boldsymbol{Z})$ is denoted by $\boldsymbol{\eta}$ and we let η_j represent its j-th element. We define a set $I_j := \{k : |\eta_k| = |\eta_j|\}$, which will be used in characterization of gradients. We also define an inverse mapping for ranking of indices: $\sigma : \{1, \ldots, p\} \to \{1, \ldots, p\}$ such that $\sigma(i) = j$ representing $|\boldsymbol{\eta}|_{(i)} = |\eta_j|$. Consider a toy example $\boldsymbol{\eta} = (-2, -4, 3, 1)$, then the ranking of magnitudes is $(3, 1, 2, 4)$ whose inverse gives: $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 1$ and $\sigma(4) = 4$. We state the following theorem to give a concrete form of gradients $\partial\tau/\partial\alpha_i$, which is used in the projected gradient descent (PGD) in Algorithm 1.

Theorem 1 *The gradients satisfy*

$$\frac{\partial \tau}{\partial \alpha_i} = \mathbb{E}\frac{1}{|I_{\sigma(i)}|D(\boldsymbol{\alpha}, \tau)}\sum_{j \in I_{\sigma(i)}}(\eta_j - \beta_j)\,\mathrm{sign}(\eta_j)\tau \tag{2.4}$$

where $D(\boldsymbol{\alpha}, \tau)$ is a negative constant independent of index i.

The form of $D(\boldsymbol{\alpha}, \tau)$ in the denominator and the proof of Theorem 1 can be found in Appendix B.1, where we also show that $D(\boldsymbol{\alpha}, \tau)$ is always negative. In practice, we can either set the step size s_t as constant or simply set $D = 1$ to save computation time. We remark that, using a constant step size and $\mathbb{E}\left(\sum_{j \in I_{\sigma(i)}}(\eta_j - \beta_j)\,\mathrm{sign}(\eta_j)\tau\right)/|I_{\sigma(i)}|$ as the gradient is equivalent to using a time-dependent $s_t = s \cdot D(\boldsymbol{\alpha}^t, \tau_t)$ and the actual gradient $\frac{\partial\tau}{\partial\alpha_i}$.

2.2. Projection onto non-negative decreasing vectors

We notice that $\boldsymbol{\alpha}$ (and $\boldsymbol{\lambda}$) must be non-negative and decreasing, hence the vanilla gradient descent unsuitable for this constrained optimization problem of $\boldsymbol{\alpha}$. Let $\mathcal{S}$ denotes the set of non-negative and decreasing vectors in $\mathbb{R}^p$ (i.e. $\boldsymbol{\alpha} \in \mathcal{S} \Rightarrow \alpha_i \geq \alpha_{i+1} \geq 0, \forall i$). Define the projection on to $\mathcal{S}$ as

$$\Pi_{\mathcal{S}}(\boldsymbol{\gamma}) = \mathrm{argmin}_{\boldsymbol{\gamma}' \in \mathcal{S}}\frac{1}{2}\|\boldsymbol{\gamma} - \boldsymbol{\gamma}'\|_2^2. \tag{2.5}$$

We propose to use projected gradient descent (PGD) to solve the optimization problem with constraint in $\mathcal{S}$, which is illustrated in the following section. To compute the projection onto $\mathcal{S}$, we propose a novel algorithm (Algorithm 3 `ProjectOnS()` in appendix B.2) and prove its correctness in a corresponding theorem (Theorem 2). At a high level, the algorithm iteratively finds the increasing sub-sequences of $\boldsymbol{\gamma}$ and averages them until the sequence becomes non-increasing.

2.3. Projected Gradient Descents

Now that we have the gradients in Theorem 1 and the projection in Algorithm 3, the projected gradient descent, shown in Algorithm 1, is straight-forward:

Algorithm 1: `Projected Gradient Descent` (PGD)

```
Input: initial $\boldsymbol{\alpha}^0$, step size $\{s_t\}$
for $t = 1, \cdots, T$ do
    $\boldsymbol{\gamma}^{t+1} = \boldsymbol{\alpha}^t - s_t \nabla_{\boldsymbol{\alpha}}(\tau(\boldsymbol{\alpha}^t))$
    $\boldsymbol{\alpha}^{t+1} = \texttt{ProjectOnS}(\boldsymbol{\gamma}^{t+1})$
end for
Output: $\boldsymbol{\alpha}^T$
```

We highlight that Algorithm 1 is only one form of PGD. In fact, with a concrete form of the gradients, we can use any off-the-shelf first-order optimizers to find $\boldsymbol{\alpha}$ iteratively. Some examples include projected versions of Heavy Ball method [14], Nesterov accelerated gradient descent [13], Adam [11], etc.

To better understand the convergence behavior of PGD, we note that the domain $\mathcal{S}$ is clearly convex and we need to study the convexity of the objective function τ. Unfortunately, $\tau(\boldsymbol{\alpha})$ for SLOPE is in general non-convex: even in the Lasso AMP regime, it is shown that $\tau(\alpha)$ is only a quasi-convex function of α [12, Theorem 3.3]. We note that some non-convex problems may enjoy desirable properties such as having unique global minimum or local minima do not exist. As for SLOPE, the analysis on quasi-convexity of $\tau(\alpha)$ has not been well established but in practice, we do not observe any local minimum.

Remarkably, the gradient information that we use distinguishes our work from [10]. We pause a bit and compare our approach with theirs, as they work under the same (asymptotic) assumptions as our AMP regime. Instead of optimizing directly on τ, they propose to optimize the η in the functional space [10, Proposition 3]: for a fixed candidate τ, they use the finite approximation with 2048 grids to solve a functional optimization, whose minimum is $\mathcal{L}(\tau)$. Next, they check the feasibility of the candidate τ by whether $\mathcal{L}(\tau) \leq \delta(\tau^2 - \sigma_w^2)$. Lastly, a binary search is conducted to find the optimal τ (smallest feasible τ) and the optimal design can be derived from the corresponding η. In summary, this approach took a detour of by a zero-order optimization algorithm, as the authors did not search over $\boldsymbol{\lambda}$ (or $\boldsymbol{\alpha}$) directly. Our first-order algorithm overcomes the seemingly unwieldy computation burden, especially in the high dimension when p is very large.

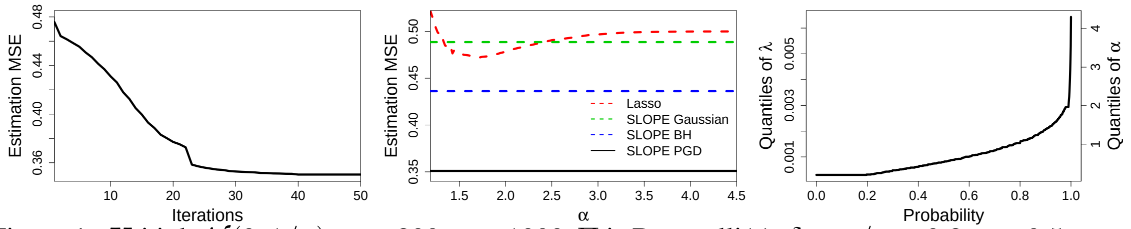

Figure 1: $\boldsymbol{X}$ i.i.d. $\mathcal{N}(0, 1/n), n = 300, p = 1000, \Pi$ is Bernoulli(ϵ), $\delta = n/p = 0.3, \epsilon = 0.5, \sigma_w = 0$. Left: PGD iteratively finds minimizing $\boldsymbol{\alpha}$. Middle: Red dashed line is LASSO MSE path; other lines are different SLOPE MSE. Right: Best $\boldsymbol{\alpha}$ (or $\boldsymbol{\lambda}$) sequence found by PGD.

Once we find the desirable $\boldsymbol{\alpha}$, the calibration (2.1) allows us to convert $\boldsymbol{\alpha}$ to $\boldsymbol{\lambda}$ in the original SLOPE problem. We demonstrate in Figure 1, SLOPE can outperform the best-tuned Lasso significantly. In Figure 1, SLOPE reduces $\mathtt{MSE}(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$ from 0.473 by Lasso and to 0.350 by SLOPE, a 26% drop in the estimation error.

3. k-level SLOPE

We propose to approximate the SLOPE problem by restricting that the penalty $\boldsymbol{\lambda}$ only contains k different non-negative values, which we denote by $(\lambda_1, \cdots, \lambda_k; S_1, \cdots, S_{k-1})$. Here λ_i denotes the penalty magnitude and S_i represents the splitting index in $[p]$, where the penalty magnitudes change, i.e, $S_i - S_{i-1}$ entries in $\boldsymbol{\lambda}$ take the value λ_i. We note that λ_i is decreasing in i while S_i is increasing, guaranteeing that $\boldsymbol{\lambda}$ satisfies the assumption of SLOPE. As an example in $\mathbb{R}^5$, $\boldsymbol{\lambda} = (7, 5, 1; 2, 3) = (7, 7, 5, 1, 1)$. We name this restricted SLOPE problem as the *k-level SLOPE* and design the $(2k-1)$ degree of freedom penalty $\boldsymbol{\lambda}$ respectively.

Note that k-level SLOPE is always a sub-case of $(k+1)$-level SLOPE. Therefore, by allowing k to take values other than 1 and p, we can trade off the difficulty of designing the penalty and the accuracy gain by employing more penalty levels. We demonstrate that empirically, the trade-off is surprisingly encouraging: even 2 or 3 levels of penalty sequence can exploit the benefit of SLOPE.

3.1. Practical penalty design for k-level SLOPE

We emphasize that in the general regime beyond AMP and CGMT, we cannot access the gradient information nor the functional optimization in [10] for two reasons: the true $\boldsymbol{\beta}$ distribution is not known in real data and the data matrix $\boldsymbol{X}$ is too restrict (i.i.d. with a specific variance). To design the k-level SLOPE penalty in the real-world datasets, we propose the Coordinate Descent (CD, Algorithm 2)[3] and compare to the PGD in Algorithm 1 under the AMP and CGMT regimes.

Algorithm 2: Coordinate Descent (CD)

```
Input: initial $\boldsymbol{\lambda}$, $\mathtt{MSE}_{old} = \infty$
while $\mathtt{MSE} < \mathtt{MSE}_{old}$ do
    Set $\mathtt{MSE}_{old} = \mathtt{MSE}$
    for $i = 1, \ldots, k$ do
        $\lambda_i = \text{argmin}_{\lambda_i \in (\lambda_{i+1}, \lambda_{i-1})} \mathtt{MSE}(\{\lambda_j\}; \{S_j\})$
    end for
    for $i \in \{1, \ldots, k-1\}$ do
        $S_i = \text{argmin}_{S_i \in (S_{i-1}, S_{i+1})} \mathtt{MSE}(\{\lambda_j\}; \{S_j\})$
    end for
end while
Output: $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_k; S_1, \cdots, S_{k-1})$
```

We highlight some details of Algorithm 2 that make it efficient and practical. First of all, Algorithm 2 directly works on $\boldsymbol{\lambda}$ instead of $\boldsymbol{\alpha}$ (the calibration is generally unavailable). Second, the projection is not needed as in Algorithm 1 since $\boldsymbol{\lambda}$ is decreasing and non-negative by our definition of the search domain. Third, Algorithm 2 is flexible in the following sense: (1) we can choose any order of coordinates to successively minimize the error, e.g. by $\lambda_1, S_1, \lambda_2, S_2, \cdots$; (2) we can use the grid search or binary search for the magnitudes and splits. To further demonstrate the utility of k-level SLOPE in practice, we apply the model to real datasets, where $\mathtt{MSE}(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$ is intractable, and focus on the prediction $\mathtt{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$.

3. We slightly abuse the notation of $\mathtt{MSE}$ to mean either the estimation error (only available in synthetic data) or the prediction error, when it is clear from context.

4. Experiment

In this experiment, we compare the performance of 2-level SLOPE with Lasso in linear regression setting. The dataset we adopt is atherosclerosis cardiovascular disease (ASCVD), which records medical information of 236 patients and their corresponding ASCVD risk score (outcome variable). We select 1000 features out of 4216 features, which has the largest correlation with the outcome variable. We conduct 20-fold cross-validation and calculate the cross-validation prediction $\texttt{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$. For the sake of implementation consistency, we adopt R package `SLOPE` to run both Lasso and SLOPE in experiments. Using grid search, the optimal prediction $\texttt{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ given by Lasso is 0.528. The optimal prediction $\texttt{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ given by 2-level SLOPE (using Algorithm 2) is 0.489.

We further extend the idea of k-level SLOPE in logistic regression and justify the results on Alzheimer's Disease Neuroimaging Initiative (ADNI) gene dataset. The dataset contains over 19000 genomic features of 649 patients, along with a binary disease status (normal or ill). We select the first 300 patients in the original dataset and 500 features out of the total features, which has the largest correlation with the outcome variable. We conduct 10-fold cross-validation and calculate the cross-validation prediction accuracy. Using grid search, the optimal prediction accuracy given by Lasso is 0.62, while 2-level SLOPE (using Algorithm 2) achieves optimal prediction accuracy 0.66.

5. Discussion

In this work, we propose a framework to flexibly and efficiently design the SLOPE penalty sequence. Under the AMP setting, our first-order PGD approach is capable of finding the effective penalty sequence with reasonable computation budget. In the practical world beyond the AMP setting, we propose k-level SLOPE, which empirically provides decent results. Much room is left for future study. From a theoretical perspective, the quasi-convexity of $\tau(\alpha)$ in AMP setting is still not being well studied. If the quasi-convexity indeed holds true for SLOPE AMP, then we can guarantee that the minimizing $\boldsymbol{\lambda}$ is indeed the global minimizer and thus claim our design is optimal. From a practical perspective, we anticipate that k-level SLOPE can also be explored in various applications that adopts the Lasso, such as the matrix completion, the compressed sensing and the neural network regularization.

References

[1] Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57(2): 764–785, 2011.

[2] Mohsen Bayati and Andrea Montanari. The lasso risk for gaussian matrices. *IEEE Transactions on Information Theory*, 58(4):1997–2017, 2011.

[3] Pierre C Bellec, Guillaume Lecué, Alexandre B Tsybakov, et al. Slope meets lasso: improved oracle bounds and optimality. *The Annals of Statistics*, 46(6B):3603–3642, 2018.

[4] Małgorzata Bogdan, Ewout Van Den Berg, Chiara Sabatti, Weijie Su, and Emmanuel J Candès. Slope—adaptive variable selection via convex optimization. *The annals of applied statistics*, 9(3):1103, 2015.

[5] Zhiqi Bu, Jason Klusowski, Cynthia Rush, and Weijie Su. Algorithmic analysis and statistical estimation of slope via approximate message passing. In *Advances in Neural Information Processing Systems*, pages 9361–9371, 2019.

[6] Zhiqi Bu, Jason Klusowski, Cynthia Rush, and Weijie Su. Algorithmic analysis and statistical estimation of slope via approximate message passing. *arXiv preprint arXiv:1907.07502*, 2019.

[7] Michael Celentano, Andrea Montanari, and Yuting Wei. The lasso with general gaussian designs with applications to hypothesis testing. *arXiv preprint arXiv:2007.13716*, 2020.

[8] David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[9] David L Donoho, Arian Maleki, and Andrea Montanari. Message passing algorithms for compressed sensing: I. motivation and construction. In *2010 IEEE information theory workshop on information theory (ITW 2010, Cairo)*, pages 1–5. IEEE, 2010.

[10] Hong Hu and Yue M Lu. Asymptotics and optimal designs of slope for sparse linear regression. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 375–379. IEEE, 2019.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Ali Mousavi, Arian Maleki, Richard G Baraniuk, et al. Consistent parameter estimation for lasso and approximate message passing. *The Annals of Statistics*, 46(1):119–148, 2018.

[13] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pages 543–547, 1983.

[14] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[15] Weijie Su, Emmanuel Candes, et al. Slope is adaptive to unknown sparsity and asymptotically minimax. *The Annals of Statistics*, 44(3):1038–1068, 2016.

[16] Christos Thrampoulidis, Samet Oymak, and Babak Hassibi. The gaussian min-max theorem in the presence of convexity. *arXiv preprint arXiv:1408.4837*, 2014.

[17] Christos Thrampoulidis, Samet Oymak, and Babak Hassibi. Regularized linear regression: A precise analysis of the estimation error. *Proceedings of Machine Learning Research*, 40: 1683–1709, 2015.

[18] Christos Thrampoulidis, Ehsan Abbasi, and Babak Hassibi. Precise error analysis of regularized m-estimators in high dimensions. *IEEE Transactions on Information Theory*, 64(8): 5592–5628, 2018.

[19] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[20] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1): 49–67, 2006.

[21] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[22] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

Appendix A. Assumptions in Approximate Message Passing regime for SLOPE

Five assumptions are made in AMP regime for analysis of SLOPE [5]:

- The measurement matrix $\boldsymbol{X}$ has independent and identically-distributed (i.i.d.) gaussian entries that have mean 0 and variance $1/n$.
- The signal $\boldsymbol{\beta}$ has elements that are i.i.d. B, with $\mathbb{E}\left(B^2 \max\{0, \log B\}\right) < \infty$.
- The noise $\boldsymbol{\omega}$ is elementwise i.i.d. W, with $\sigma_w^2 := \mathbb{E}\left(W^2\right) < \infty$.
- The vector $\boldsymbol{\lambda}(p) = (\lambda_1, \ldots, \lambda_p)$ is elementwise i.i.d. Λ, with $\mathbb{E}\left(\Lambda^2\right) < \infty$.
- The ratio n/p reaches a constant $\delta \in (0, \infty)$ in the large system limit, as $n, p \to \infty$.

Appendix B. Analysis of Projected Gradient Descent for α

B.1. Proof of Theorem 1

Proof

Minimizing the estimation error is equivalent to minimizing τ. Since the AMP algorithms are working on the finite dimension, we analyze the finite-size approximation of the state evolution [6, Equation (2.5)]:

$$\tau^2 = \sigma_w^2 + \frac{1}{\delta p}\mathbb{E}\left\|\text{prox}_{J_{\boldsymbol{\alpha}\tau}}(\boldsymbol{\beta} + \tau \boldsymbol{Z}) - \boldsymbol{\beta}\right\|^2$$

In finite dimensions, the expectation is taken with respect to $\boldsymbol{Z}$. Differentiating both sides of the state evolution with respect to α_i and denoting $\tau' = \frac{\partial \tau}{\partial \alpha_i}$ give:

$$\begin{aligned} 2\tau\tau' &= \frac{\partial}{\partial \alpha_i}\left(\sigma_w^2 + \frac{1}{\delta p}\mathbb{E}\|\operatorname{prox}_{J_{\boldsymbol{\alpha}\tau}}(\boldsymbol{\beta} + \tau\boldsymbol{Z}) - \boldsymbol{\beta}\|^2\right) \\ &= \frac{1}{n}\frac{\partial}{\partial \alpha_i}\sum_{j=1}^{p}\mathbb{E}\left([\text{prox}_{J_{\boldsymbol{\alpha}\tau}}(\boldsymbol{\beta} + \tau\boldsymbol{Z})]_j - \boldsymbol{\beta}_j\right)^2 \end{aligned} \tag{B.1}$$

Recall η_j represents the j-th element of $\boldsymbol{\eta} := \text{prox}_{J_{\boldsymbol{\alpha}\tau}}(\boldsymbol{\beta} + \tau\boldsymbol{Z})$. By chain rule

$$\begin{aligned} 2\tau\tau' &= \frac{2}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j - \beta_j)\frac{\partial \eta_j}{\partial \alpha_i} \\ &= \frac{2}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j - \beta_j)\left[\sum_{k=1}^{p}\frac{d\eta_j}{da_k}\frac{\partial a_k}{\partial \alpha_i} + \frac{d\eta_j}{db_k}\frac{\partial b_k}{\partial \alpha_i}\right] \end{aligned} \tag{B.2}$$

where we define $a_k := \beta_k + \tau Z_k, b_k := \alpha_k \tau$. To calculate the derivatives, we pause to discuss forms of general derivatives of $\boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{b})$. Define

$$\partial_1 \boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{b}) := \text{diag}\left[\frac{\partial}{\partial a_1}, \frac{\partial}{\partial a_2}, \ldots, \frac{\partial}{\partial a_p}\right]\boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{b}) \tag{B.3}$$

$$\partial_2 \boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{b}) := \text{diag}\left[\frac{\partial}{\partial b_1}, \frac{\partial}{\partial b_2}, \ldots, \frac{\partial}{\partial b_p}\right]\boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{b}). \tag{B.4}$$

According to [15, Proof of Fact 3.4] and [6, Proof of Theorem 1], we have

$$[\partial_1 \boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_j = \frac{1}{\#\{1 \le k \le p : |[\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_k| = |[\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_j|\}}$$

and that

$$\frac{d}{da_k}[\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_j = \mathbb{I}\{|\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})|_j = |\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})|_k\}\,\mathrm{sign}(\boldsymbol{\eta}_j\boldsymbol{\eta}_k)[\partial_1\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_j$$

for the derivative regardng the first variable. Recall that the permutation $\sigma : \{1,\dots,p\} \to \{1,\dots,p\}$ is the inverse mapping for ranking of indices such that $|\boldsymbol{\eta}|_{(i)} = |[\boldsymbol{\eta}]_{\sigma(i)}|$. Similarly, according to [6, Proof of Theorem 1]:

$$\begin{aligned}\frac{d}{db_k}[\eta(\boldsymbol{a},\boldsymbol{b})]_j &= -\,\mathrm{sign}([\eta(\boldsymbol{a},\boldsymbol{b})]_{\sigma(k)})\frac{d}{da_{\sigma(k)}}[\boldsymbol{\eta}(\boldsymbol{a},\boldsymbol{b})]_j \\ &= \mathbb{I}\big\{|\eta(\boldsymbol{a},\boldsymbol{b})|_j = |\eta(\boldsymbol{a},\boldsymbol{b})|_{\sigma(k)}\big\}\,\mathrm{sign}\left(\eta_j\right)\left[\partial_1\eta(\boldsymbol{a},\boldsymbol{b})\right]_j. \end{aligned} \tag{B.5}$$

In addition to I_j defined in Section 2, we let $K_j := \{k : |\eta_{\sigma(k)}| = |\eta_j|\}$. We can rewrite (B.2) as

$$\begin{aligned}2\tau\tau' &= \frac{2}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j-\beta_j)\left[\sum_{k\in I_j}\frac{d\eta_j}{da_k}\frac{\partial a_k}{\partial\alpha_i} + \sum_{k\in K_j}\frac{d\eta_j}{db_k}\frac{\partial b_k}{\partial\alpha_i}\right] \\ &= \frac{2}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)\left[\frac{1}{|I_j|}\sum_{k\in I_j}\mathrm{sign}(\eta_k)\frac{\partial a_k}{\partial\alpha_i} - \frac{1}{|K_j|}\sum_{k\in K_j}\frac{\partial b_k}{\partial\alpha_i}\right] \\ &= \frac{2}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)\left[\frac{1}{|I_j|}\sum_{k\in I_j}\mathrm{sign}(\eta_k)Z_k\tau' - \frac{1}{|K_j|}\sum_{k\in K_j}(\alpha_k\tau' + \mathbb{I}\{k=i\}\tau)\right]\end{aligned}$$

Merging the terms containing the derivative τ' on one side gives

$$\begin{aligned}&\frac{1}{n}\sum_{j\in I_{\sigma(i)}}\mathbb{E}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)/|K_j| \\ &= \frac{1}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)\left[\frac{1}{|I_j|}\sum_{k\in I_j}\mathrm{sign}(\eta_k)Z_k\tau' - \frac{1}{|K_j|}\sum_{k\in K_j}\alpha_k\tau'\right] - \tau\tau'\end{aligned}$$

Notice that $|I_j| = |K_j|$ due to σ being a permutation, we can simplify above as

$$\frac{\partial\tau}{\partial\alpha_i} = \mathbb{E}\frac{1}{|I_{\sigma(i)}|D(\boldsymbol{\alpha},\tau)}\sum_{j\in I_{\sigma(i)}}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)\tau \tag{B.6}$$

where $D(\boldsymbol{\alpha},\tau)$ in the denominator is

$$D(\boldsymbol{\alpha},\tau) = -n\tau + \sum_{j=1}^{p}\mathbb{E}\frac{1}{|I_j|}(\eta_j-\beta_j)\,\mathrm{sign}(\eta_j)\sum_{k\in I_j}(\mathrm{sign}(\eta_k)Z_k - \alpha_{\sigma^{-1}(k)})$$

We next show that $D(\boldsymbol{\alpha}, \tau)$ is always negative. Firstly observe from (2.2) that

$$\tau^2 > \frac{1}{n}\sum_{j=1}^{p}\mathbb{E}(\eta_j - \beta_j)^2 \tag{B.7}$$

Now for the set I_i with a fixed index i,

$$\sum_{j\in I_i}(\eta_j - \beta_j)^2 \geq \frac{1}{|I_i|}(\sum_{j\in I_i}|\eta_j - \beta_j|)^2 \tag{B.8}$$

$$\geq \frac{1}{|I_i|}(\sum_{j\in I_i}(\eta_j - \beta_j)\operatorname{sign}(\eta_j))^2 \tag{B.9}$$

$$= \frac{1}{|I_i|}\sum_{j\in I_i}(\eta_j - \beta_j)\operatorname{sign}(\eta_j)\sum_{k\in I_i}\tau Z_k\operatorname{sign}(\eta_k) - \alpha_{\sigma^{-1}(k)}\tau \tag{B.10}$$

$$\geq \frac{\tau}{|I_i|}\sum_{j\in I_i}(\eta_j - \beta_j)\operatorname{sign}(\eta_j)\sum_{k\in I_j}Z_k\operatorname{sign}(\eta_k) - \alpha_{\sigma^{-1}(k)} \tag{B.11}$$

This in turn implies that

$$\sum_{j=1}^{p}(\eta_j - \beta_j)^2 = \sum_{j=1}^{p}\frac{1}{|I_j|}\sum_{k\in I_j}(\eta_k - \beta_k)^2 \tag{B.12}$$

$$\geq \sum_{j=1}^{p}\frac{\tau}{|I_j|}(\eta_j - \beta_j)\operatorname{sign}(\eta_j)\sum_{k\in I_j}Z_k\operatorname{sign}(\eta_k) - \alpha_{\sigma^{-1}(k)} \tag{B.13}$$

Combining with (B.7) yields $D < 0$.

■

B.2. Characterization of projection on $\mathcal{S}$

To calculate projection $\Pi_{\mathcal{S}}$, we propose the following algorithm.

Algorithm 3: `ProjectOnS` ($\Pi_{\mathcal{S}}$)

```
Input: Arbitrary sequence $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$
for $i = 1, \cdots, p$ do
    Identify the shortest sub-sequence $\{\gamma_j, \ldots, \gamma_i\}$ whose average is smaller than its left neighbor
    (with $\gamma_0 = \infty$):
        $\frac{1}{i-j+1}\sum_{k=j}^{i}\gamma_k \leq \gamma_{j-1}$
    Assign the average value to such sub-sequence for $(\gamma_j, \ldots, \gamma_i)$:
        $\gamma_j, \ldots, \gamma_i \leftarrow \frac{1}{i-j+1}\sum_{k=j}^{i}\gamma_k$
end for
Output: $\max\{\boldsymbol{\gamma}, 0\}$    (Element-wise truncation)
```

We now state that Algorithm 3 indeed find the projection onto $\mathcal{S}$:

Theorem 2 *Given* $\forall \boldsymbol{\gamma} \in \mathbb{R}^p$ *as input, Algorithm 3 outputs the projection of* $\boldsymbol{\gamma}$ *on* $\mathcal{S}$*, that is,* $\Pi_{\mathcal{S}}(\boldsymbol{\gamma})$*.*

The proof consists of two parts. In the first part we provide a detailed characterization of $\Pi_{\mathcal{S}}(\gamma)$ by partitioning the index sequence $\{1,\dots,p\}$ into a number of carefully selected sub-sequences. We prove that within each sub-sequence, $\Pi_{\mathcal{S}}(\gamma)$ takes the same value at each index, and such value is exactly the average of the sub-sequence $\boldsymbol{\gamma}$'s values at these indices. In the second part, we prove that Algorithm 1 indeed finds such sub-sequences and thus operates in a way that matches the goal of the projection $\Pi_{\mathcal{S}}(\gamma)$. The final truncation of the averaged sequence at 0 is a trivial method to guarantee the non-negativity.

We firstly prove that Algorithm 3 indeed finds the projection. To do so we firstly provide a detailed characterization of the projection, then prove that the output of Algorithm 3 matches the form of projection. We start by defining *blocks* and *segmentation blocks*, upon which our proof highly relies. Suppose $\boldsymbol{\gamma} = \{\gamma_1,\dots,\gamma_p\}$, *blocks* are subsequences defined as $B(\boldsymbol{\gamma},u) := \{\gamma_u,\dots,\gamma_{u+L(\boldsymbol{\gamma},u)-1}\}$ where length $L(\boldsymbol{\gamma},u)$ is defined as

$$L(\boldsymbol{\gamma},u) = \begin{cases} L^* & \text{if } L^* \neq \emptyset \\ p & \text{otherwise} \end{cases} \tag{B.14}$$

where

$$L^* \triangleq \min\left\{1 \le L \le p-u \,\middle|\, \forall 0 \le k \le p-u-L, \frac{1}{k+1}\sum_{i=0}^{k}\gamma_{u+L+i} < \frac{1}{L}\sum_{i=0}^{L-1}\gamma_{u+i}\right\}$$

Roughly speaking, $L(\boldsymbol{\gamma},u)$ is the minimum value of a finite set (truncated at p when the set is empty). For each element L in this set, the average value in sequence $\{\gamma_u,\dots,\gamma_{u+L-1}\}$ is always larger than that of arbitrary sequence $\{\gamma_{u+L},\dots,\gamma_{u+L+k}\}$ whose left start is γ_{u+L}. With such definition of blocks, we can now segment $\boldsymbol{\gamma}$ into $q \le p$ blocks:

$$\begin{aligned} \boldsymbol{\gamma} &= \{B(\boldsymbol{\gamma},1), B(\boldsymbol{\gamma},L(\boldsymbol{\gamma},1)+1), B(\boldsymbol{\gamma},L(\boldsymbol{\gamma},L(\boldsymbol{\gamma},1)+1)+L(\boldsymbol{\gamma},1)+1),\dots\} \\ &\triangleq \{B_1,\dots,B_q\} \end{aligned}$$

We call $B_1,\dots,B_q$ *segmentation blocks* for vector $\boldsymbol{\gamma}$. It's straightforward to see that $B_k = B(\boldsymbol{\gamma},L_k)$ where L_k satisfies $L_1 = L(\boldsymbol{\gamma},1)$ and

$$L_k = L(\boldsymbol{\gamma}, \sum_{i=1}^{k-1} L_i + 1)$$

Our result shows that for input vector γ, its projection vector $\Pi_{\mathcal{S}}(\gamma)$ takes identical values inside each of the segmentation blocks. Before formally stating the theorem, We first highlight the following fact that will be frequently used in the proof of the theorem.

Fact 1 For two sequences of length p: $\{a_i\}$ and $\{b_i\}$, if $\sum a_i = \sum b_i$, then function $g(C) := \sum(b_i - a_i + C)^2$ is monotonically increasing with respect to $|C|$.

Proof Notice that

$$\begin{aligned} \sum(b_i - a_i + C)^2 &= \sum(b_i - a_i)^2 + \sum 2C(b_i - a_i) + pC^2 \\ &= pC^2 + \sum(b_i - a_i)^2 \end{aligned}$$

Hence $g(C)$ is is monotonically increasing with respect to $|C|$. ■

Theorem 3 *Let B denote the segmentation block that contains γ_i, then*

$$(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \max\left\{\frac{1}{|B|}\sum_{\gamma_j\in B}\gamma_j, 0\right\}$$

Proof

The proof consists of two steps. In the first step, we prove that for each segmentation block B, the projection of each coordinates share the same value. That is, $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \mathcal{C}(B)$ as long as $\gamma_i \in B$. In the second step, we show that this constant is the mean of the block truncated at 0: $\mathcal{C}(B) = \max\left\{\frac{1}{|B|}\sum_{\gamma_j\in B}\gamma_j, 0\right\}$.

Step 1 Without loss of generality, we consider $B = B(\boldsymbol{\gamma}, u)$. We know from definition of blocks that $\forall 1 \leq l \leq L-1$, $\exists k_l$ s.t. $\frac{1}{k_l}\sum_{i=1}^{k_l}\gamma_{u+l-1+i} \geq \frac{1}{l}\sum_{i=1}^{l}\gamma_{u+i-1}$. We use induction to prove that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l}$, $\forall 1 \leq l \leq L(\gamma, u) - 1$. For $l = 1$, assume $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+1}$. Consider two cases: (i) $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > \gamma_u$. (ii) $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u \leq \gamma_u$. We now show that both cases lead to contradiction and hence do not hold. In case (i), we consider

$$(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \begin{cases}\max\{\gamma_u, (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+1}\} & \text{if } i = u\\ (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i & \text{otherwise}\end{cases}$$

then obviously,

$$\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_u - \gamma_u\right| < |(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u - \gamma_u|$$

which leads to that $\frac{1}{2}\|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2 < \frac{1}{2}\|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2$. This contradicts to the definition of projection. In case (ii), from definition of blocks we have that $\exists k_0 \geq 1$ s.t. $\frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+i} \geq \gamma_u$. Consider

$$(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \begin{cases}(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u & \text{if } i \in \{u+1, \ldots, u+k_0\}\\ (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i & \text{otherwise}\end{cases}$$

Notice that $\frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+i} \geq \gamma_u \geq (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+1}$, we have for $i \in \{u+1, \ldots, u+k_0\}$, $\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i\right|$ is a constant independent of i and that

$$\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+i}\right| < \left|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+i}\right|$$

According to Fact B.2, we define substitution for $i \in \{u+1, \ldots, u+k_0\}$: $b_i = \frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+i}$, $a_i = \gamma_{u+i}$, $b_i + C_1 = (\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i$ and $b_i + C_2 = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i$. Then since $|C_1 < C_2|$, we have $\frac{1}{2}\|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2 < \frac{1}{2}\|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2$, which contradicts to the definition of projection.

Now assume the statement holds for $1 \leq l \leq l_0 - 1$, that is $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u = \cdots = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0-1}$, we want to prove that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}$. Since the projection is on $\mathcal{S}$, by definition we know $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u$ can never be smaller than $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}$. We now assume $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}$

and consider two cases: (i) $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i}$. (ii) $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u \leq \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i}$. To complete the proof, it suffices for us to show that neither of the cases can hold without contradictions. In case (i), we consider

$$(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \begin{cases} \max\{\frac{1}{l_0}\sum_{j=0}^{l_0-1}\gamma_{u+j}, (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}\} \\ \qquad\qquad \text{if } i \in \{u, \dots, u+l_0-1\} \\ (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i \qquad\qquad \text{otherwise} \end{cases}$$

then obviously for $i \in \{u, \dots, u+l_0-1\}$, $\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i\right|$ is a constant independent of i and that

$$\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i}\right| < \left|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i}\right|$$

According to Fact B.2, using the same substitution as that in analysis of $l = 1$, we have that $\frac{1}{2}\|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2 < \frac{1}{2}\|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2$, which makes contradiction to the definition of projection. In case (ii), from definition of blocks we have that $\exists k_0 \geq 1$ s.t. $\frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+l_0-1+i} \geq \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i}$. Now we consider

$$(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \begin{cases} (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u & \text{if } i \in \{u+l_0, \dots, u+l_0-1+k_0\} \\ (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i & \text{otherwise} \end{cases}$$

Notice that $\frac{1}{k_0}\sum_{i=1}^{k_0}\gamma_{u+l_0-1+i} \geq \frac{1}{l_0}\sum_{i=0}^{l_0-1}\gamma_{u+i} \geq (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}$, we have for $i \in \{u+l_0, \dots, u+l_0-1+k_0\}$, $\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i\right|$ is a constant independent of i and that

$$\left|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{k_0}\sum_{i=0}^{k_0-1}\gamma_{u+l_0+i}\right| < \left|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i - \frac{1}{k_0}\sum_{i=0}^{k_0-1}\gamma_{u+l_0+i}\right|$$

Again according to Fact B.2, we have $\frac{1}{2}\|(\widetilde{\Pi}_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2 < \frac{1}{2}\|(\Pi_{\mathcal{S}}(\boldsymbol{\gamma})) - \boldsymbol{\gamma}\|_2^2$, which contradicts to the definition of projection. This implies that it can never happen that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u > (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+l_0}$, which completes the induction. We have proved that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_u = \cdots = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_{u+L(\boldsymbol{\gamma},u)-1} \overset{\Delta}{=} \mathcal{C}(B(u))$ for each segmentation block $B(u)$ of vector $\boldsymbol{\gamma}$.

Step 2 Now we already know that inside each segmentation block, the projection of each coordinate is a constant $\mathcal{C}(B)$, we now optimize the sequence $\{\mathcal{C}(B_i)\}_{i=1}^{q}$. According to Fact B.2, inside each B_i, the optimal constant (i.e. constant gives smallest ℓ_2 error $\text{argmin}_{C\geq 0}\frac{1}{2}\sum_{\gamma_j \in B_i}(\gamma_j - C)^2$) is : $\max\left\{\frac{1}{|B_i|}\sum_{\gamma_j\in B_i}\gamma_j, 0\right\}$. Meanwhile, it's feasible to set

$$(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i = \max\left\{\frac{1}{|B|}\sum_{\gamma_j\in B}\gamma_j, 0\right\}$$

since we have that $\max\left\{\frac{1}{|B_i|}\sum_{\gamma_j\in B_i}\gamma_j, 0\right\} \geq \max\left\{\frac{1}{|B_{i+1}|}\sum_{\gamma_j\in B_{i+1}}\gamma_j, 0\right\}$ by definition of blocks. This wraps up the proof.

■

B.3. Proof of Theorem 2

We next prove the validity of Algorithm 3.

Proof Suppose $\boldsymbol{\gamma}$ has segmentation blocks $B_1, \dots, B_q$, we firstly prove that $(\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_i = (\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i$ for $i \leq |B_1|$. We let $\gamma_j(t)$ denote the value of γ_j at the moment i was assigned from t to $t+1$ in Algorithm 3 (i.e. the time when first t iterations are finished). We also let $\gamma_j(0)$ denote the initial value of γ_j in the input. Then clearly $(\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_j = \max\{\gamma_j(p), 0\}$. During the value-averaging step, the algorithm is constantly transporting values from elements with larger index to those with smaller. Hence it's straightforward to see that

$$\sum_{j=1}^{J} \gamma_j(t) \geq \sum_{j=1}^{J} \gamma_j(t-1) \tag{B.15}$$

for arbitrary $J, t \in \{1, \dots, p\}$. First assume $\gamma_1(p) = \dots = \gamma_{\widetilde{L}_1}(p) > \gamma_{\widetilde{L}_1+1}(p)$. Since Algorithm 3 only involves averaging values among subsequences, we have that $\sum_{j=1}^{p} \gamma_j(p) = \sum_{j=1}^{p} \gamma_j$. Moreover since $\gamma_{\widetilde{L}_1}(p) > \gamma_{\widetilde{L}_1+1}(p)$, there's no value-averaging steps between any one of the first $\widetilde{L}_1$ elements and one of the rest elements. This implies

$$\sum_{j=1}^{\widetilde{L}_1} \gamma_j(p) = \sum_{j=1}^{\widetilde{L}_1} \gamma_j \tag{B.16}$$

By definition of blocks, we know that $\exists k$ such that $\frac{1}{k}\sum_{i=1}^{k} \gamma_{\widetilde{L}_1+i} \geq \frac{1}{\widetilde{L}_1}\sum_{i=1}^{\widetilde{L}_1} \gamma_i = \gamma_1(p)$. By (B.15) we have that

$$\frac{1}{k}\sum_{i=1}^{k} \gamma_{\widetilde{L}_1+i} \leq \frac{1}{k}\sum_{i=1}^{k} \gamma_{\widetilde{L}_1+i}(p) \leq \gamma_{\widetilde{L}_1+1}(p)$$

Together with above, this implies that $\gamma_1(p) \leq \gamma_{\widetilde{L}_1+1}(p)$, which contradicts to the assumption. Hence we have that $\widetilde{L}_1 \geq L_1$.

On the other hand, if $\widetilde{L}_1 > L_1$, then at the moment i is assigned to be $\widetilde{L}_1 + 1$ in the algorithm (i.e. the time when first $\widetilde{L}_1$ iterations are finished), we must have that

$$\frac{\sum_{j=1}^{\widetilde{L}_1} \gamma_j(\widetilde{L}_1 - 1)}{\widetilde{L}_1} \geq \frac{\sum_{j=1}^{L_1} \gamma_j(\widetilde{L}_1 - 1)}{L_1}$$

This implies that

$$\frac{\sum_{j=L_1+1}^{\widetilde{L}_1} \gamma_j(\widetilde{L}_1 - 1)}{\widetilde{L}_1 - L_1} \geq \frac{\sum_{j=1}^{L_1} \gamma_j(\widetilde{L}_1 - 1)}{L_1} \tag{B.17}$$

By (B.15) we have

$$\frac{\sum_{j=1}^{L_1} \gamma_j}{L_1} \leq \frac{\sum_{j=1}^{L_1} \gamma_j(\widetilde{L}_1 - 1)}{L_1} \tag{B.18}$$

Meanwhile at $t = \widetilde{L}_1 - 1$, the sum of first L_1 terms is the same as that in $\boldsymbol{\gamma}$. This implies

$$\begin{aligned}\sum_{j=L_1+1}^{\widetilde{L}_1} \gamma_j(\widetilde{L}_1 - 1) &= \sum_{j=1}^{L_1} \gamma_j + \sum_{j=L_1+1}^{\widetilde{L}_1} \gamma_j - \sum_{j=1}^{L_1} \gamma_j(\widetilde{L}_1 - 1) \\ &\leq \sum_{j=L_1+1}^{\widetilde{L}_1} \gamma_j\end{aligned} \tag{B.19}$$

where the last inequality is given by (B.15). Combining (B.17), (B.18) and ((B.19)) yields

$$\frac{\sum_{j=L_1+1}^{\widetilde{L}_1} \gamma_j}{\widetilde{L}_1 - L_1} \geq \frac{\sum_{j=1}^{L_1} \gamma_j}{L_1}$$

This contradicts to definition of L_1 in (B.14). Hence we have that $\widetilde{L}_1 = L_1$. This means $\gamma_1(p) = \cdots = \gamma_{L_1}(p) > \gamma_{L_1+1}(p)$. Recall that $(\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_j = \max\{\gamma_j(p), 0\}$, this together with (B.16) yields

$$\begin{aligned}(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_1 &= \max\left\{\frac{1}{|B_1|}\sum_{j=1}^{L_1} \gamma_j, 0\right\} = (\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_1 \\ &= \cdots = (\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_{L_1} > (\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_{L_1+1}\end{aligned}$$

Now we have prove that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i = (\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_i$ for $i \leq |B_1|$ and that there is no interaction between element in B_1 and that outside B_1. This implies that the existence of B_1 does *not* affect the rest of output values $(\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_{i>|B_1|}$. Hence we can ignore B_1 and repeat exactly the same procedure to prove that $(\Pi_{\mathcal{S}}(\boldsymbol{\gamma}))_i = (\Lambda_{\mathcal{S}}(\boldsymbol{\gamma}))_i$ when $|B_1| + 1 \leq i \leq |B_2|$ and that there is no interactions between element in B_2 and that outside B_2. Iteratively we can prove $\Pi_{\mathcal{S}}(\boldsymbol{\gamma}) = \Lambda_{\mathcal{S}}(\boldsymbol{\gamma})$

■