圧縮センシングアルゴリズムの構造学習における構造パラメータの二値化

指導教員

飯國 洋二 教授

報告者

今井 智也

大阪大学 基礎工学部 システム科学科

2024年2月8日

概要

圧縮センシングは，劣決定系の線形逆問題において解のスパース性（ほとんどの成分が 0 であるという性質）を仮定できる場合に，そのスパース性を利用して解を推定するための数学的な枠組みである．圧縮センシングのための代表的なアルゴリズムの一つである ISTA（Iterative Shrinkage Thresholding Algorithm）を基盤として，アルゴリズムの構造自体をデータから学習する AS-ISTA（Architecture Searched-ISTA）が提案されている．AS-ISTA では ISTA の 2 つのステップの重み付き和を新たな更新式として考え，その重みの値を構造パラメータによって定義する．構造パラメータを適切に学習することで推定精度と収束速度の面で ISTA よりも優れた性能を有するが，2 つのステップに対する重みが一般にはどちらも非負の実数となるため，更新式の解釈が不透明になる可能性がある．本研究では，AS-ISTA の重み部分を二値化し，どちらのステップを採用したかを明確にする手法を提案する．また，二値化によって収束速度と復元精度を維持したまま反復アルゴリズムの計算時間を削減できることを示す．

目 次

第 1 章　序論 **1**

第 2 章　研究背景 **3**
2.1　圧縮センシング [1] 3
2.2　ℓ_1 再構成 3
2.3　圧縮センシングアルゴリズム 3
2.3.1　近接勾配法 4
2.3.2　ISTA(Iterative Shrinkage Thresholding Algorithm) . . . 4
2.4　深層展開によるパラメータ学習 6
2.4.1　深層展開 6
2.4.2　LISTA（Learned ISTA） 6
2.5　ニューラルネットワークのアーキテクチャ探索 7
2.5.1　NAS（Neural Architecture Search） 8
2.5.2　DARTS（Differentiable Architecture Search） 8
2.6　AS-ISTA(Architecture Searched-ISTA) 9

第 3 章　提案手法 **11**
3.1　重みの設定 11
3.2　パラメータの学習方法 12

第 4 章　計算機シミュレーション **14**
4.1　シミュレーション設定 14
4.2　シミュレーション結果 14
4.2.1　初期値パターン 1 14
4.2.2　初期値パターン 2 18

第 5 章　結論 **23**

謝辞 **25**

第1章　序論

未知ベクトルをその次元より少ない次元の線形観測から復元する問題は，信号処理における基本的な問題設定の一つである．この問題は，画像処理におけるノイズ除去や超解像，音声処理におけるノイズ除去や音源分離など，様々な分野で応用されている．このような線形逆問題は劣決定系と呼ばれ，解が無数に存在して一意に定まらない．劣決定線形逆問題の 1 つである圧縮センシング [1] では，解がスパースである（非零成分）ことを仮定し，そのスパース性を活用することで復元を行う．

圧縮センシングのための手法の 1 つとして，ℓ_1 ノルムを用いた LASSO（Least Absolute Shrinkage and Selection Operator）[2] がある．ℓ_1 ノルムは凸ではあるが微分可能ではないので，微分可能性を仮定した単純な勾配法は適用できない．このように微分可能ではない目的関数を持つ最適化問題に対して，近接勾配法 [3, 4] というアルゴリズムが提案されている．近接勾配法は，微分可能な項に関する勾配降下と微分不可能な項に関する近接作用素を用いて凸最適化問題を反復的に解く手法である．LASSO に対して近接勾配法を適用したアルゴリズムは ISTA（Iterative Shrinkage Thresholding Algorithm）[5] と呼ばれる．ISTA はその反復式に現れるステップサイズと呼ばれるパラメータを適切に設定することで，LASSO の最適解を求めることができる．

ISTA の収束速度を向上させるための方法として，深層展開 [6] と呼ばれるアプローチが提案されている．深層展開では，反復的な構造を持つ信号処理アルゴリズムの信号の流れを時間方向に展開した信号流グラフを考える．それをフィードフォワード型のニューラルネットワークとみなし，深層学習技術を適用することでアルゴリズム内のパラメータを調整する．ISTA に深層展開を適応して得られたアルゴリズムは LISTA（Learned ISTA）[6] と呼ばれ，元の ISTA よりもよい特性を達成することが示されている．学習するパラメータや損失関数などを適切に設定することにより，反復構造を持つ信号処理アルゴリズムの収束速度や推定速度を向上させることが期待できる．

機械学習分野において，深層ニューラルネットワークのアーキテクチャを自動で設計する NAS（Neural Architecture Search）[7] が研究されている．NAS の手法の一つに DARTS（Differentiable Architecture Search）[8] がある．DARTS はニューラルネットワークの構造を連続空間において最適化する手法であり，微分可能なソフトマックス関数を用いることで，構造の効率的な探索を行うことができる．

DARTS のアイデアを ISTA に適用した手法として AS-ISTA（Architecture Searched-ISTA）[9] が提案されている．従来の ISTA では勾配降下ステップと縮小ステップを交互に作用させる形であるが，AS-ISTA では二つのステップの

重み付き和を考え，その重みの値を構造パラメータを用いて定義する．重みは構造パラメータのソフトマックス関数となっており，深層展開によってステップサイズパラメータと構造パラメータが交互に学習される．AS-ISTA は従来の深層展開でパラメータを学習させた ISTA よりも高い推定精度と収束速度を達成することが実験的に示されている．しかし勾配降下ステップと縮小ステップの重みの値は 0 より大きく 1 より小さい実数値であるため，一般には更新式の解釈が不透明になる可能性がある．また，重みの値がともに 0 より大きい場合，計算量が元の ISTA の約 2 倍となる．

本研究では，AS-ISTA の重みの値が 0 または 1 になるように二値化をする手法（BAS-ISTA： Binarized AS-ISTA）を提案する．従来のソフトマックス型とは異なる重みを用いて，インクリメンタル学習のくり返しが終わるごとに構造パラメータの二値化処理を入れることにより，AS-ISTA の重みの値を二値化する．これにより，どちらのステップを採用したかを明確になり，反復式を計算する際にかかる計算時間が削減される．計算機シミュレーションによって，BAS-ISTA は AS-ISTA と同等の推定精度を達成することを示す．また，復元に必要な計算時間を元の ISTA と比べて最大 48.2 %削減できることを示す．

本論文では，まず第 2 章で圧縮センシングの問題設定と最適化問題について述べて，ISTA について述べる．その後深層展開によるパラメータの学習について述べ，ネットワークの構造最適化を行う NAS と DARTS について説明する．そして DARTS を ISTA に適応した AS-ISTA について説明する．第 3 章では，AS-ISTA の構造パラメータを二値化する手法である BAS-ISTA について説明する．第 4 章では，BAS-ISTA の計算機シミュレーションの問題設定を述べ，AS-ISTA との比較結果を示す．第 5 章では，本論文の結果と今後の課題について述べる．

第2章　研究背景

本章ではまず圧縮センシングの問題設定に関して説明を行い，次に圧縮センシングのための最適化問題について述べる．そしてその最適化問題を解く手法として，近接勾配法に基づいたアルゴリズムである ISTA を紹介する．

2.1　圧縮センシング [1]

原信号 $\boldsymbol{x} \in \mathbb{R}^N$ は非零要素数が N に対して十分に小さいスパースなベクトルであると仮定する．観測ベクトル $\boldsymbol{y} \in \mathbb{R}^M (M < N)$ は

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \tag{2.1}$$

と与えられる．ここで $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ は既知の観測行列，$\boldsymbol{w} \in \mathbb{R}^M$ は観測雑音である．圧縮センシングでは，与えられた観測ベクトル $\boldsymbol{y}$ から原信号 $\boldsymbol{x}$ を可能な限り高い精度で復元することを目標とする．

2.2　ℓ_1 再構成

式 (2.1) では $M < N$ であるため，観測雑音がなかったとしても $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ を満たす $\boldsymbol{x}$ は無数に存在して一意には定まらない．しかし，解に対する事前情報を利用することで，一意に解を定めることができる．このアプローチは正則化と呼ばれ，圧縮センシングではできるだけ非零成分が少ない解を求めることになる．また，観測 $\boldsymbol{y}$ には雑音 $\boldsymbol{w}$ が含まれるので，$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ は正確には成り立たない．以上より，観測雑音を含む線形観測においてスパースな解を得るための最適化問題として，以下の ℓ_1 再構成問題が提案されている．

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{R}^N}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \right\}. \tag{2.2}$$

これは LASSO（Least Absolute Shrinkage and Selection Operator）[2] とも呼ばれる．$\lambda > 0$ はこの二つの項のうち，どちらの項を重要視するかを調整するための正則化係数である．λ が大きくなると式 (2.2) の正則化項の影響が強くなり，解はスパースとなりやすくなる．

2.3　圧縮センシングアルゴリズム

本節ではまず近接勾配法について説明し，最適化問題 (2.2) を解くためのアルゴリズムである ISTA について説明する．

Algorithm 1 近接勾配法

Input: $\boldsymbol{x}^0$

Output: $\boldsymbol{x}^T$

1: Fix $\gamma_t > 0$ $(t = 0, 1, \ldots, T-1)$.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: $\quad \boldsymbol{x}^{(t+1)} = \mathrm{prox}_{\gamma_t g}\left(\boldsymbol{x}^{(t)} - \gamma_t \nabla f\left(\boldsymbol{x}^{(t)}\right)\right)$
4: **end for**

2.3.1 近接勾配法

最適化問題 (2.2) の目的関数は微分可能ではないので単純な勾配法を適用することはできない．式 (2.2) のような最適化問題を解くための手法の一つとして近接勾配法 [3, 4] が提案されている．

近接勾配法では，次の形の凸最適化問題を考える．

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad \{g(\boldsymbol{x}) + h(\boldsymbol{x})\}. \tag{2.3}$$

ここで，$g(\boldsymbol{x})$ は微分可能であり，その勾配 $\nabla g(\boldsymbol{x})$ がリプシッツ連続であるとする．すなわち，ある定数 $\tilde{L} > 0$ が存在して任意の $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ に対して

$$\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{y})\| \leq \tilde{L}\|\boldsymbol{x} - \boldsymbol{y}\| \tag{2.4}$$

が成り立つとする．

近接勾配法の更新式は，

$$\boldsymbol{r}^{(t)} = \boldsymbol{x}^{(t)} - \lambda \nabla g(\boldsymbol{x}^{(t)}), \tag{2.5}$$

$$\boldsymbol{x}^{(t+1)} = \mathrm{prox}_{\lambda h}(\boldsymbol{r}^{(t)}) \tag{2.6}$$

で与えられる．ここで，λ はステップサイズであり，prox は h の近接作用素である．近接作用素は，

$$\mathrm{prox}_{\lambda h}(\boldsymbol{v}) = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\left(h(\boldsymbol{x}) + \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2\right) \tag{2.7}$$

と定義される．近接勾配法のアルゴリズムを Algorithm 1 に示す．

2.3.2 ISTA(Iterative Shrinkage Thresholding Algorithm)

最適化問題 (2.2) に対する近接勾配法のアルゴリズムは ISTA（Iterative Shrinkage Thresholding Algorithm）[5] と呼ばれる．ISTA はスパースな解を求めるための手法として広く用いられる．

まず，

$$g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{2.8}$$

$$h(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1 \tag{2.9}$$

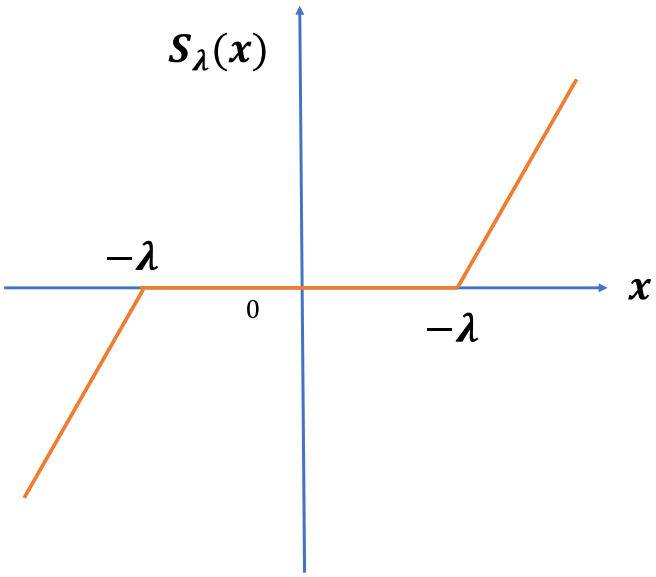


図 2.1: ソフト閾値作用素の概形

とおくと，最適化問題 (2.2) は式 (2.3) の問題に帰着される．関数 $g(\boldsymbol{x})$ の勾配は

$$\nabla g(\boldsymbol{x}) = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) \tag{2.10}$$

であり，$h(\boldsymbol{x})$ の近接写像を計算すると

$$\mathrm{prox}_{\lambda h}(\boldsymbol{x}) = \mathrm{S}_\lambda(\boldsymbol{x}) \tag{2.11}$$

となる．ただし，$\mathrm{S}_\lambda(\boldsymbol{x})$ はソフト閾値作用素

$$\mathrm{S}_\lambda(x) = \begin{cases} x - \lambda & (x \geq \lambda) \\ 0 & (-\lambda < x < \lambda) \\ x + \lambda & (x \leq -\lambda) \end{cases} \tag{2.12}$$

である．ただし，$\lambda > 0$，$x \in \mathbb{R}$ である．概形は図 2.1 のようになる．ソフト閾値作用素の入力がベクトルの場合は，ベクトルの各成分に対して式 (2.12) の関数を作用させるものとする．

以上より，ISTA の反復アルゴリズムは以下のようになる．

$$\boldsymbol{r}^{(t)} = \boldsymbol{x}^{(t)} - \alpha\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y}), \tag{2.13}$$

$$\boldsymbol{x}^{(t+1)} = \mathrm{S}_{\alpha\lambda}(\boldsymbol{r}^{(t)}). \tag{2.14}$$

ISTA の反復アルゴリズムを Algorithm 2 に示す．ステップサイズ α を

Algorithm 2 Iterative Shrinkage-Thresholding Algorithm (ISTA)

Input: $\boldsymbol{x}^0$ $\alpha \in (0, \frac{1}{\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})})$.
Output: $\boldsymbol{x}^{(t)}$.
1: **while** 停止条件を満たすまで **do**
2: $\quad \boldsymbol{r}^{(t)} = \boldsymbol{x}^{(t)} - \alpha \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y})$
3: $\quad \boldsymbol{x}^{(t+1)} = \mathrm{S}_{\alpha\lambda}(\boldsymbol{r}^{(t)})$
4: $\quad t \leftarrow t + 1$
5: **end while**

$$0 < \alpha < \frac{1}{\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})} \tag{2.15}$$

というように選べば発散せずに最適解が得られる．ここで $\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})$ は行列 $\boldsymbol{A}^\top \boldsymbol{A}$ の最大固有値である．

2.4 深層展開によるパラメータ学習

ISTA は最適なステップサイズを選ぶことで必ず最適解へと収束することが保証されているが，その収束速度はステップサイズの設定値によって異なるため，各反復ごとに適切なステップサイズを設定することが望ましい．その手法の一つに深層展開という手法が提案されている．

2.4.1 深層展開

深層展開 [6] は，従来の最適化アルゴリズムの信号流グラフを時間方向に展開してフィードフォワード型のニューラルネットワークとみなすことにより，深層学習技術を適用してアルゴリズム内のパラメータを学習する手法である．従来の最適化反復アルゴリズム図 2.2（左）の反復処理過程を時間方向に展開して得られる信号流グラフを図 2.2（右）に示す．展開後の信号流グラフの構造はフィードフォワード型のニューラルネットワークによく似ているため，サブプロセス A，B，C がパラメータに関して微分可能で非零の導関数値を持つならば，深層学習技術（誤差逆伝播法や確率的勾配降下法など）をこの信号流グラフに適用することができる．このようにしてアルゴリズム中に表れる調節可能なパラメータをミニバッチ学習で調節することができる．

2.4.2 LISTA（Learned ISTA）

LISTA [6] では，深層展開のコンセプトに基づいて ISTA の各反復をニューラルネットワークの層としてモデル化する．そして ISTA の中で登場する線形推定式の推定行列を学習パラメータとする．ISTA の勾配降下ステップ (2.14) は

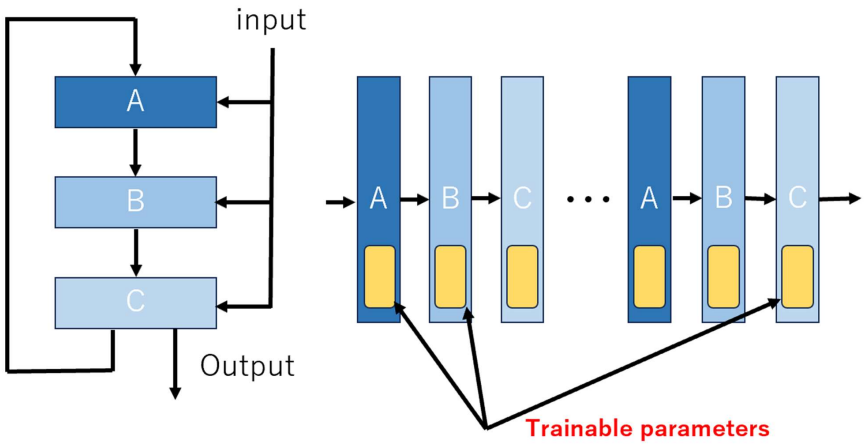


図 2.2: （左）反復アルゴリズムの信号流グラフ,（右）時間方向に展開された信号流グラフ

$\boldsymbol{x}^{(t)}$ と $\boldsymbol{y}$ を線形変換して得られるベクトルの和となっている．LISTA は ISTA の勾配降下ステップを変更した

$$\boldsymbol{r}^{(t)} = \boldsymbol{B}^{(t)}\boldsymbol{x}^{(t)} + \boldsymbol{S}^{(t)}\boldsymbol{y}, \tag{2.16}$$

$$\boldsymbol{x}^{(t+1)} = \mathrm{S}_{\alpha\lambda}(\boldsymbol{r}^{(t)}) \tag{2.17}$$

という形の反復式を持つ．ここで α と行列 $\boldsymbol{B}^{(t)} \in \mathbb{R}^{N\times N}$，行列 $\boldsymbol{S}^{(t)} \in \mathbb{R}^{N\times M}$ が学習可能パラメータであるため，学習過程において調整可能なパラメータとなっている．LISTA は ISTA に対して大幅な収束速度の向上を与えることが実験的に示されている．

2.5　ニューラルネットワークのアーキテクチャ探索

機械学習の分野では最適なニューラルネットワークの構造やハイパーパラメータを自動的に見つける手法として NAS（Neural Architecture Search）がある．本節では NAS について説明し，NAS の手法の一つである DARTS（Differentiable Architecture Search）について説明する．

2.5.1 NAS（Neural Architecture Search）

NAS [7] は機械学習モデル，特に深層ニューラルネットワークの最適なアーキテクチャを自動的に探索する手法である．このアプローチは，手動でのネットワーク設計に代わるものとして，大きな注目を集めている．NAS の主な目的は，特定のタスクに最適化されたネットワーク構造を自動的に生成することである．これにより，人間の専門家が行う時間を要する試行錯誤のプロセスを短縮し，人間が見落とす可能性のある創造的な設計を発見することができる．しかしながら，膨大な計算を並列で行うために計算コストが大きくなり，ネットワークの内部構造が理解しづらく，解釈性に欠けることが課題として挙げられる．

2.5.2 DARTS（Differentiable Architecture Search）

効率よく NAS を実行する手法の一つとして DARTS [8] が提案されている．DARTS は，ネットワークアーキテクチャの選択を連続的な空間上で表現することにより，探索プロセスを微分可能にすることで，勾配降下法を用いて最適なアーキテクチャを探索できるようにする．

DARTS では，ニューラルネットワークはセルという構造ブロックから構成される．各セルは有向非巡回グラフ（Directed acyclic graph; DAG）を使用して表現される．異なる演算を組み合わせたニューラルネットワークのアーキテクチャを効率的に検索する．

DARTS においては，ニューラルネットワークモデルを DAG として以下のように定義する．

$$x^{(i)} = \sum_{j<i} o^{(i,j)}(x^{(j)}). \tag{2.18}$$

ここで，$o^{(i,j)}$ はニューラルネットワークに含まれる畳み込み層，プーリング層などの演算に対応し，$x^{(i)}$ はその入出力にあたるノードを表す．DARTS ではこのような演算の重み付き和

$$\hat{o}^{(i,j)} = \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o^{(i,j)} \tag{2.19}$$

を考え，その重みパラメータを学習することで演算の選択を行う．ここで，$\mathcal{O}$ は可能な演算の集合であり，$\alpha_o^{(i,j)}$ は演算 o の重みである．

DARTS の学習アルゴリズムを Algorithm 3 に示す．まず各ノード間 (i,j) で α をパラメータとした混合演算 $\hat{o}^{(i,j)}$ を作成する．次にニューラルネットワークの重み w と構造パラメータ α を，確率的勾配降下法により交互に更新する．ただし，過学習を防ぐために勾配を計算する損失関数が異なっており，w は訓練データ，α は検証データをそれぞれ用いる．η_w，η_α はそれぞれ w と α の学習率である．最後に構造パラメータが最も大きい演算を選択して，ネットワークの構造を決定する．

Algorithm 3 Differentiable Architecture Search (DARTS)

1: 各ノード間 (i,j) で α をパラメータとした混合演算 $\hat{o}^{(i,j)}$ を作成.
2: **while** 停止条件を満たすまで **do**
3: $\quad w \leftarrow w - \eta_w \nabla_w L_{\text{train}}(w, \alpha)$
4: $\quad \alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha L_{\text{val}}(w, \alpha)$
5: **end while**
6: $\hat{o}^{(i,j)} \leftarrow o^{(i,j)} = \underset{o \in \mathcal{O}}{\text{argmax}} \quad \alpha_o^{(i,j)}$ for each edge(i,j)

2.6 AS-ISTA(Architecture Searched-ISTA)

従来の深層展開アルゴリズムでは，多くの場合アルゴリズムの構造自体は固定である．しかしその構造が最適解の探索に適しているかどうかは定かではない．そこで DARTS の演算選択を ISTA に適用し，深層展開における構造探索を実現する方法が提案されている [9].

AS-ISTA では，反復式

$$\boldsymbol{r}^{(t)} = w_{x,f}^{(t)} f(\boldsymbol{x}^{(t)}) + w_{x,g}^{(t)} g(\boldsymbol{x}^{(t)}), \tag{2.20}$$

$$\boldsymbol{x}^{(t+1)} = w_{r,f}^{(t)} f(\boldsymbol{r}^{(t)}) + w_{r,g}^{(t)} g(\boldsymbol{r}^{(t)}) \tag{2.21}$$

を考える．候補演算 f, g はそれぞれの勾配降下ステップと縮小ステップであり，

$$f(\boldsymbol{x}^{(t)}) = \boldsymbol{x}^{(t)} - \alpha \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y}), \tag{2.22}$$

$$g(\boldsymbol{x}^{(t)}) = S_{\alpha\lambda}(\boldsymbol{x}^{(t)}) \tag{2.23}$$

である．ここで，重み $w_{x,f}^{(t)}, w_{x,g}^{(t)}, w_{r,f}^{(t)}, w_{r,g}^{(t)}$ は DARTS と同様にして，

$$w_{x,f}^{(t)} = \frac{\exp(\beta_1^{(t)})}{\exp(\beta_1^{(t)}) + \exp(\beta_2^{(t)})}, w_{x,g}^{(t)} = \frac{\exp(\beta_2^{(t)})}{\exp(\beta_1^{(t)}) + \exp(\beta_2^{(t)})}, \tag{2.24}$$

$$w_{r,f}^{(t)} = \frac{\exp(\gamma_1^{(t)})}{\exp(\gamma_1^{(t)}) + \exp(\gamma_2^{(t)})}, w_{r,g}^{(t)} = \frac{\exp(\gamma_2^{(t)})}{\exp(\gamma_1^{(t)}) + \exp(\gamma_2^{(t)})} \tag{2.25}$$

とする．また $\beta_1^{(t)}, \beta_2^{(t)}, \gamma_1^{(t)}, \gamma_2^{(t)}$ はそれぞれの演算の重みを表す構造パラメータである．

AS-ISTA の学習アルゴリズムを Algorithm 4 に示す．学習方法にはインクリメンタル学習 [10, 11] を用いている．インクリメンタル学習では，ネットワークの総数 T の値を 1 から始め，そこからミニバッチごとに最終層の時点での損失関数の値を計算する．そして内ループにおいて所定の回数パラメータが更新されると，外のループにおいて T の値を最大値 $T_{\max}$ まで 1 ずつインクリメンタルする．インクリメンタル学習を行うことで勾配消失問題を防ぐことができると考えられる．またモデルの最適化に Adam アルゴリズム [12] を用いている．Adam はその適応的な学習率調整機能により，個々のパラメータに対して効率的な更新を行うことが可能である．N_{b} をミニバッチサイズとし，用意し

Algorithm 4 AS-ISTA(Architecture Searched-ISTA)

Input: 初期パラメータ $\alpha, \theta = (\beta_1^{(t)}, \beta_2^{(t)}, \gamma_1^{(t)}, \gamma_2^{(t)})$ と Adam オプティマイザーのパラメータを設定する.

for $T = 1, \ldots, T_{\max}$ **do**
  for $i = 1, \ldots, n$ **do**
    T 層のネットワークでの出力を $\hat{x}$, 損失関数を $L_{\text{train}}(\alpha, \theta)$ とする.
    $\alpha \leftarrow \text{Adam}(\alpha, \nabla_\alpha L_{\text{train}}(\alpha, \theta))$
  end for
  for $i = 1, \ldots, n$ **do**
    T 層のネットワークでの出力を $\hat{x}$, 損失関数を $L_{\text{val}}(\alpha, \theta)$ とする.
    $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta L_{\text{train}}(\alpha, \theta))$
  end for
end for

た N_{b} 個のデータ $(x_i, y_i)_{1 \leq i \leq N_{\text{b}}}$ に対して，勾配計算に用いる損失関数はともに以下のような出力 $\hat{x}_i$ と真値 x_i の MSE(平均二乗誤差) を用いる．すなわち

$$L_{\text{train}}(\alpha, \theta) = L_{\text{val}}(\alpha, \theta) = \frac{1}{N_{\text{b}}} \sum_{i=1}^{N_{\text{b}}} (\hat{x}_i - x_i)^2 \tag{2.26}$$

である．初期値は $\beta_1^{(t)} = 10, \beta_2^{(t)} = -10, \gamma_1^{(t)} = -10, \gamma_2^{(t)} = 10$ として学習を行う．これは AS-ISTA の反復式 (2.20)(2.21) が ISTA の反復式 (2.13)(2.14) と一致するようにするためである.

第3章　提案手法

AS-ISTA では，勾配降下ステップと縮小ステップの重みの値にソフトマックス型を用いているため，0 より大きく 1 より小さい実数値をとる. その結果更新式の解釈が不透明である可能性がある．

本研究では，インクリメンタル学習の最中に重みの二値化をおこなうことで二値化構造に適したステップサイズパラメータを学習する手法を提案する．これは AS-ISTA で得られた重みを学習後に閾値を用いて二値化をしてもステップサイズパラメータ α は元の構造を最適にするために学習されているため新しい構造に対してはうまくいかないためである．

3.1　重みの設定

重みの二値化を行うために新たな重みを設定する．今回は 2 パターンの重みで実験を行う．式 (2.20) 式 (2.21) の重み部分について，

パターン 1 $w_{x,f}^{(t)} = \beta^{(t)}$，$w_{x,g}^{(t)} = 1 - \beta^{(t)}$，$w_{r,f}^{(t)} = 1 - \gamma^{(t)}$，$w_{r,g}^{(t)} = \gamma^{(t)}$

パターン 2 $w_{x,f}^{(t)} = \frac{\beta_1^{(t)}}{\beta_1^{(t)}+\beta_2^{(t)}}$，$w_{x,g}^{(t)} = \frac{\beta_2^{(t)}}{\beta_1^{(t)}+\beta_2^{(t)}}$，$w_{r,f}^{(t)} = \frac{\gamma_1^{(t)}}{\gamma_1^{(t)}+\gamma_2^{(t)}}$，$w_{r,g}^{(t)} = \frac{\gamma_2^{(t)}}{\gamma_1^{(t)}+\gamma_2^{(t)}}$

となるように設定した．パターン 1，パターン 2 ともに重みの和が 1 となるように設定した．これを式 (2.20) 式 (2.21) に代入すると更新式は

パターン 1

$$\boldsymbol{r}^{(t)} = \beta^{(t)} f(\boldsymbol{x}^{(t)}) + (1 - \beta^{(t)}) g(\boldsymbol{x}^{(t)}), \tag{3.1}$$

$$\boldsymbol{x}^{(t+1)} = (1 - \gamma^{(t)}) f(\boldsymbol{r}^{(t)}) + \gamma^{(t)} g(\boldsymbol{r}^{(t)}) \tag{3.2}$$

パターン 2

$$\boldsymbol{r}^{(t)} = \frac{\beta_1^{(t)}}{\beta_1^{(t)} + \beta_2^{(t)}} f(\boldsymbol{x}^{(t)}) + \frac{\beta_2^{(t)}}{\beta_1^{(t)} + \beta_2^{(t)}} g(\boldsymbol{x}^{(t)}), \tag{3.3}$$

$$\boldsymbol{x}^{(t+1)} = \frac{\gamma_1^{(t)}}{\gamma_1^{(t)} + \gamma_2^{(t)}} f(\boldsymbol{r}^{(t)}) + \frac{\gamma_2^{(t)}}{\gamma_1^{(t)} + \gamma_2^{(t)}} g(\boldsymbol{r}^{(t)}) \tag{3.4}$$

となる．

Algorithm 5 BAS-ISTA(Binarized AS-ISTA)

Input: パラメータ $\alpha, \theta = (\beta_1^{(t)}, \beta_2^{(t)}, \gamma_1^{(t)}, \gamma_2^{(t)}), w_{x,f}^{(t)}, w_{x,g}^{(t)}, w_{r,f}^{(t)}, w_{r,g}^{(t)}$ と Adam オプティマイザーのパラメータを設定する.

for $T = 1, \ldots, T_{\max}$ **do**
  for $i = 1, \ldots, n$ **do**
    T 層のネットワークでの出力を $\hat{x}$, 損失関数を $L_{\text{train}}(\alpha, \theta)$ とする.
    $\alpha \leftarrow \text{Adam}(\alpha, \nabla_\alpha L_{\text{train}}(\alpha, \theta))$
  end for
  for $j = 1, \ldots, n$ **do**
    T 層のネットワークでの出力を $\hat{x}$, 損失関数を $L_{\text{val}}(\alpha, \theta)$ とする.
    $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta L_{\text{train}}(\alpha, \theta))$
  end for
  for $k = 1, \ldots, T$ **do**
    $w_{x,f}^{(k)} \leftarrow \text{threshold\_function}(w_{x,f}^{(k)})$
    $w_{x,g}^{(k)} \leftarrow 1 - w_{x,f}^{(k)}$
    $w_{r,f}^{(k)} \leftarrow \text{threshold\_function}(w_{r,f}^{(k)})$
    $w_{r,g}^{(k)} \leftarrow 1 - w_{r,f}^{(k)}$
  end for
end for

3.2 パラメータの学習方法

パラメータの学習は Algorithm 5 に従って行う．学習時に重みが 0 以上 1 以下となるように制約をかける．学習方法はインクリメンタル学習を採用する．インクリメンタル学習の過程で追加された層に含まれる構造パラメータを，段階的に二値化していく．二値化をしながら学習を行う方法を BAS-ISTA(Binarized AS-ISTA) と呼ぶことにする．

N_{b} をミニバッチサイズとし，用意した N_{b} 個のデータ $(x_i, y_i)_{1 \leq i \leq N_{\text{b}}}$ に対して，勾配計算に用いる損失関数はともに以下のような出力 $\hat{x}_i$ と真値 x_i の MSE(平均二乗誤差) を用いる．

$$L_{\text{train}}(\alpha, \theta) = L_{\text{val}}(\alpha, \theta) = \frac{1}{N_{\text{b}}} \sum_{i=1}^{N_{\text{b}}} (\hat{x}_i - x_i)^2. \tag{3.5}$$

二値化処理では，重みが

$$\text{threshold\_function}(w) = \begin{cases} 1 & (w \geq 0.5) \\ 0 & (w < 0.5) \end{cases} \tag{3.6}$$

となるように処理を行う．パターン 1 の場合は

$$\beta^{(t)} = \begin{cases} 1 & (\beta^{(t)} \geq 0.5) \\ 0 & (\beta^{(t)} < 0.5) \end{cases} \tag{3.7}$$

$$\gamma^{(t)} = \begin{cases} 1 & (\gamma^{(t)} \geq 0.5) \\ 0 & (\gamma^{(t)} < 0.5) \end{cases} \tag{3.8}$$

となる．パターン 2 の場合は

$$\beta_1^{(t)} = \begin{cases} 1 & (\beta_1^{(t)} \geq \beta_2^{(t)}) \\ 0 & (\beta_1^{(t)} < \beta_2^{(t)}) \end{cases} \tag{3.9}$$

$$\beta_2^{(t)} = \begin{cases} 1 & (\beta_2^{(t)} \geq \beta_1^{(t)}) \\ 0 & (\beta_2^{(t)} < \beta_1^{(t)}) \end{cases} \tag{3.10}$$

$$\gamma_1^{(t)} = \begin{cases} 1 & (\gamma_1^{(t)} \geq \gamma_2^{(t)}) \\ 0 & (\gamma_1^{(t)} < \gamma_2^{(t)}) \end{cases} \tag{3.11}$$

$$\gamma_2^{(t)} = \begin{cases} 1 & (\gamma_2^{(t)} \geq \gamma_1^{(t)}) \\ 0 & (\gamma_2^{(t)} < \gamma_1^{(t)}) \end{cases} \tag{3.12}$$

となる．

ここで $w_{x,f}^{(t)} = 1$ または $w_{r,f}^{(t)} = 1$ であることは式 (2.20) 式 (2.21) において勾配降下ステップを選択することを表し，それぞれ

$$\boldsymbol{r}^{(t)} = f(\boldsymbol{x}^{(t)}) \tag{3.13}$$
$$\boldsymbol{x}^{(t+1)} = f(\boldsymbol{r}^{(t)}) \tag{3.14}$$

となる．$w_{x,g}^{(t)} = 1$ または $w_{r,g}^{(t)} = 1$ であることは式 (2.20) 式 (2.21) において縮小ステップを選択することを表し，それぞれ

$$\boldsymbol{r}^{(t)} = g(\boldsymbol{x}^{(t)}) \tag{3.15}$$
$$\boldsymbol{x}^{(t+1)} = g(\boldsymbol{r}^{(t)}) \tag{3.16}$$

となる．

第4章　計算機シミュレーション

計算機シミュレーションにより，提案手法の特性を評価する．今回は2パターンの初期値設定に対する結果を示す．AS-ISTA と 2 パターンの重み設定時の BAS-ISTA との比較結果を示す．また重みを二値化したことによって反復アルゴリズム内の計算時間が短縮したことを示す．

4.1　シミュレーション設定

学習に使用するデータセットは，ミニバッチサイズ N として，以下の式によって生成する．

$$\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{w}_i \in \mathbb{R}^M \quad (i = 1, 2, \cdots, N_{\mathrm{b}}). \tag{4.1}$$

$M = 150$，$N = 75$ とし，$\boldsymbol{A} \in \mathbb{R}^{M \times N}$ の各要素は平均 0，分散 1 の独立同分布なガウス分布に従うとする．$\boldsymbol{x}_i \in \mathbb{R}^N$ は各要素が独立同分布なガウス分布に従う乱数であり，非零成分の生起確率は 0.08 である．$\boldsymbol{w}_i \in \mathbb{R}^M$ は各要素が平均 0，分散 $\sigma_{\boldsymbol{w}}^2$ の独立同分布なガウス雑音に従う．$\boldsymbol{w}_i$ を生成するときは信号対雑音比（$\mathrm{SNR} = 10\log_{10}\left(\frac{\sigma_{\boldsymbol{x}}^2}{\sigma_{\boldsymbol{w}}^2}\right)$）が 10 dB になるように $\sigma_{\boldsymbol{w}}^2$ を設定する．ここで $\sigma_{\boldsymbol{x}}^2$ は原信号 $\boldsymbol{x}$ の非零成分の分散を表す．N_{b} はミニバッチサイズである．モデルの最適化に Adam アルゴリズムを採用した．今回の実験ではステップサイズパラメータ α の学習率は $\eta_\alpha = 7.0 \times 10^{-5}$ とした．反復回数 T は $T = 100$ とした．シミュレーションでは $\lambda = 10$ として，観測行列 $\boldsymbol{A}$ を 100 個生成し，1 つの $\boldsymbol{A}$ につき 100 個の原信号 $\boldsymbol{x}$ を生成して推定を行い，各反復ごとの MSE と目的関数の値のそれぞれの平均値を評価した．

4.2　シミュレーション結果

4.2.1　初期値パターン１

まず１つ目の初期値パターンとして，AS-ISTA のパラメータの初期値はステップサイズ $\alpha^{(t)} = \frac{1}{\lambda_{max}(\boldsymbol{A}^\top \boldsymbol{A})}$ であり，構造パラメータは $\beta_1 = 10$，$\beta_2 = 9.15$，$\gamma_1 = 9.15$，$\gamma_2 = 10$ とした．これは AS-ISTA の反復式において式 (2.20) では勾配降下ステップと縮小ステップが 7:3 の割合になるように式 (2.21) では勾配降下ステップと縮小ステップが 3:7 の割合になるように設定した．同様の割合になるようにパターン１の重み設定の場合では $\beta = 0.7$，$\gamma = 0.7$，パターン２の重み設定の場合では $\beta_1 = 0.7$，$\beta_2 = 0.3$，$\gamma_1 = 0.3$，$\gamma_2 = 0.7$ で行った．

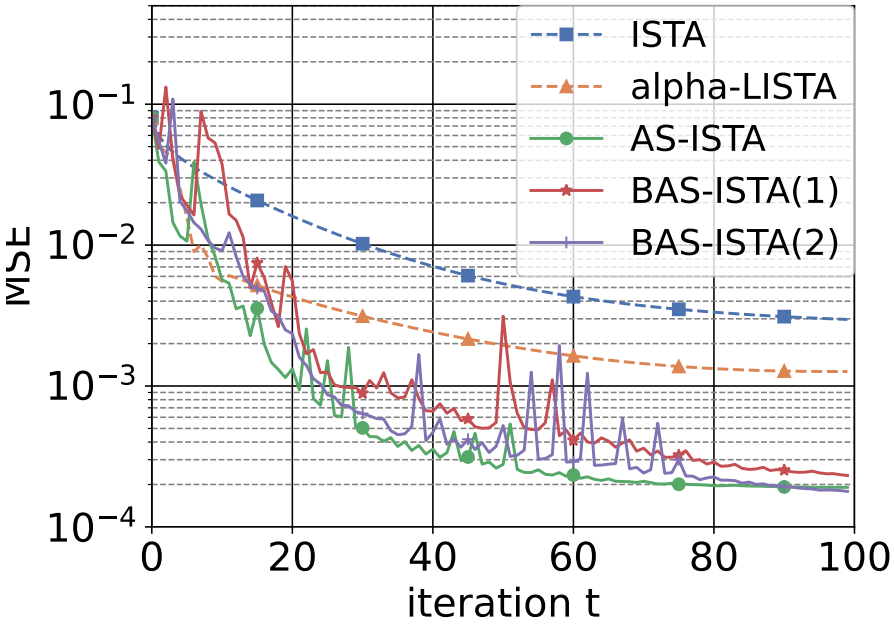


図 4.1: 各反復ごとの MSE の値

AS-ISTA の構造パラメータ θ の学習率は $\eta_\theta = 4.0 \times 10^{-3}$，パターン 1 の構造パラメータ θ の学習率は $\eta_\theta = 7.0 \times 10^{-3}$，パターン 2 の構造パラメータ θ の学習率は $\eta_\theta = 7.0 \times 10^{-2}$ とした．

図 4.1 は，ISTA・ステップサイズパラメータ α のみを学習させた α-LISTA・AS-ISTA・式 (3.1) と式 (3.2) を用いた BAS-ISTA（BAS-ISTA(1)）・式 (3.3) と式 (3.4) を用いた BAS-ISTA（BAS-ISTA (2)）の各反復ごとの MSE の値を表す．α-LISTA は ISTA よりも精度がよく，AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) は 3 つとも α-LISTA，ISTA よりも精度が優れていることがわかる．BAS-ISTA(1) と BAS-ISTA(2) は収束途中で MSE の値が大きくなっている部分がところどころあるが，最終的に 100 反復後には AS-ISTA とほぼ同様の結果を得た．このことから重みを二値化しても精度を落とすことなく推定ができたことが分かる．

図 4.2 は各反復ごとの目的関数の値を表している．BAS-ISTA(1) と BAS-ISTA(2) は 20 ステップ目までは目的関数の値が大きくなっているが，最終的には AS-ISTA と同様の結果を得た．また目的関数の大きさは AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) よりも α-LISTA，ISTA の方が小さいことがわかる．これより $\hat{\boldsymbol{x}}_i$ が真値 $\boldsymbol{x}_i$ に近づくとは限らないということが分かる．

図 4.3 は各反復ごとのステップサイズパラメータの値を表している．AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) は反復回数が増えるにつれてステップサイズパラメータの値が小さくなる傾向が見て取れる．

図 4.4a は学習後の AS-ISTA の構造パラメータの結果から，AS-ISTA が最終的にとっている構造を可視化したものである．一つ一つのブロックは AS-ISTA の $t = 0$ から $t = 99$ までの勾配降下ステップの重みを表し，左上から右下にかけて並べている．奇数列は反復式の一行目 (2.20) の勾配降下ステップの重み，偶数列は二式目 (2.21) の勾配降下ステップの重みの値に対応している．つまり，

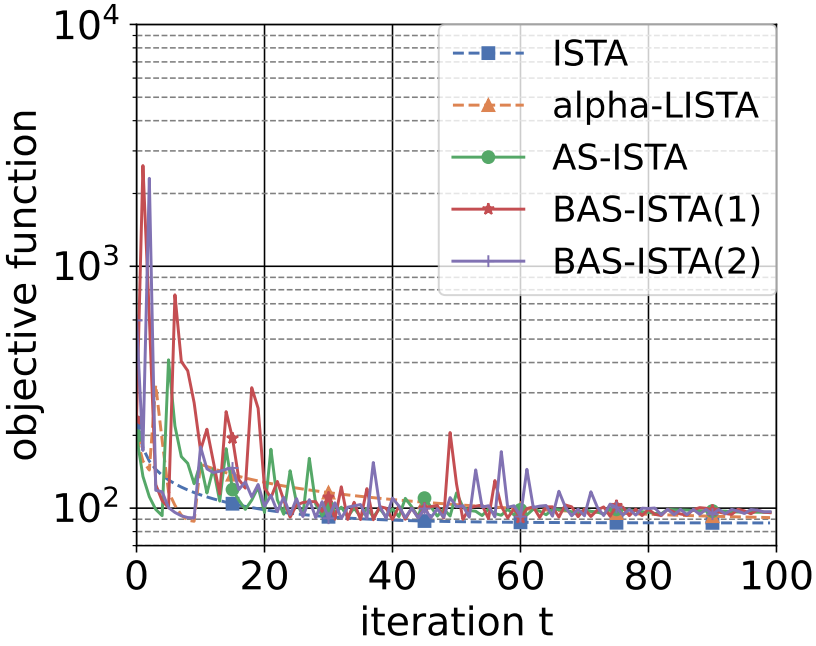


図 4.2: 各反復ごとの目的関数の値

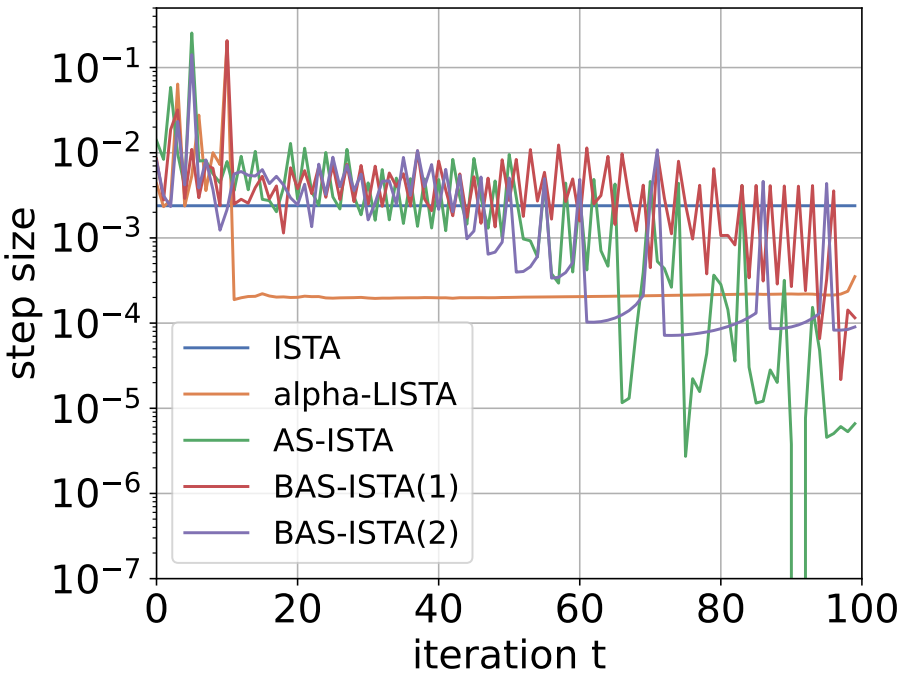
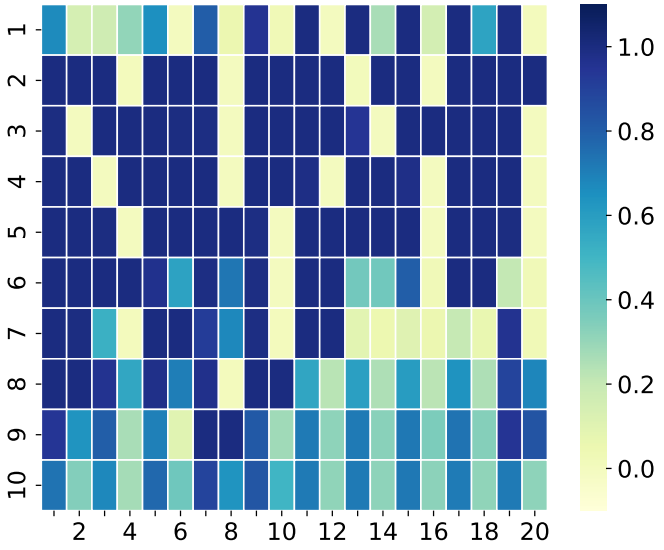
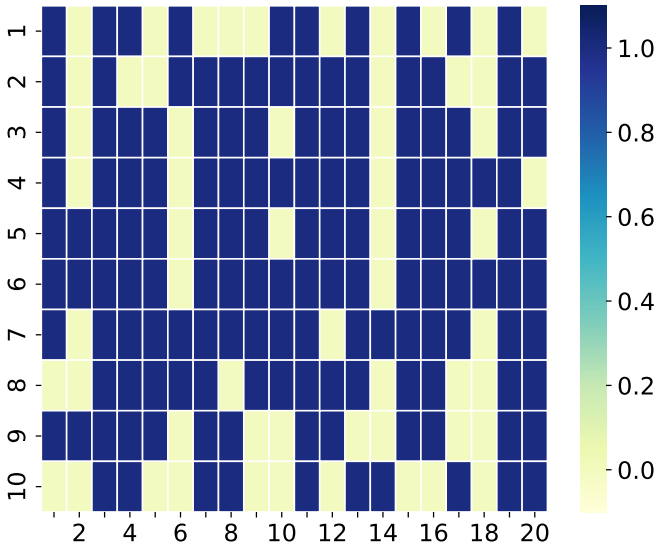


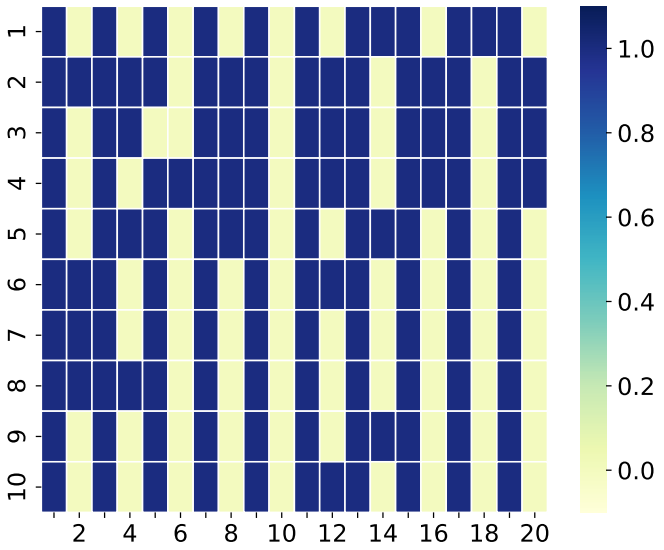
図 4.3: 反復ごとのステップサイズの値



(a) 学習後の AS-ISTA の構造



(b) 学習後の BAS-ISTA(1) の構造



(c) 学習後の BAS-ISTA(2) の構造

図 4.4: 初期値パターン 1 の構造の比較

表 4.1: 各アルゴリズムの 100 反復にかかる計算時間と AS-ISTA 比較

アルゴリズム	計算時間（秒）	AS-ISTA 比 (%)
ISTA	1.72×10^{-2}	50.7
AS-ISTA	3.39×10^{-2}	-
BAS-ISTA(1)	1.84×10^{-2}	54.3
BAS-ISTA(2)	1.83×10^{-2}	54.0

1 行目の 1，2 列目の二つのブロックは $t = 0$ における二式を表し，1 行目の 3，4 に従って行う列目の二つのブロックは $t = 1$ における二式を表している．また，各ブロックの色は重みの値を表しており，青が 1.0，黄色が 0.0 を表している．AS-ISTA の構造パラメータは 11 反復目から 50 反復目まではほぼ 1.0 または 0.0 に近い値をとっているが，それ以外では 0.5 付近の値をとっていることがわかる．

図 4.4b と図 4.4c は学習後の BAS-ISTA(1)，BAS-ISTA(2) の構造を表している．BAS-ISTA(1) と BAS-ISTA(2) は二値化処理され重み部分は 1.0 と 0.0 のみをとっている．ともに青色が多いことから縮小ステップよりも勾配降下ステップを多く選択した方が MSE の精度がよくなることがわかる．

最後に ISTA，AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) のそれぞれの 100 反復時にかかる計算時間を表 4.1 に示す．計算時間の計測は 100 回の反復を 10000 回行い，その平均値を計算した．計算には，CPU：Core i9 10900X 10core/20thread 3.7GHz・GPU：NVIDIA GeForce GTX 1650 4GB・メモリ：64GB の計算機を用いた．それぞれ計算したところ 1.72×10^{-2} 秒，3.39×10^{-2} 秒，1.84×10^{-2} 秒，1.83×10^{-2} 秒であった．これより BAS-ISTA(1) は，AS-ISTA に比べて 54.3 パーセント，BAS-ISTA(2) は AS-ISTA に比べて 54.0 パーセントの計算時間となった．

4.2.2 初期値パターン 2

2 つ目のパターンとして AS-ISTA のパラメータの初期値はステップサイズ $\alpha^{(t)} = \frac{1}{\lambda_{max}(\boldsymbol{A}^\top \boldsymbol{A})}$ であり，構造パラメータは $\beta_1 = 10$，$\beta_2 = -10$，$\gamma_1 = -10$，$\gamma_2 = 10$ とした．これは AS-ISTA の反復式において初期値の段階では ISTA と同じ構造になるように設定したものである．同様の割合になるようにパターン 1 の重み設定の場合では $\beta_1 = 1.0$，$\gamma_2 = 1.0$，パターン 2 の重み設定の場合では $\beta_1 = 1.0$，$\beta_2 = 0.0$，$\gamma_1 = 0.0$，$\gamma_2 = 1.0$ で行った．構造パラメータ θ の学習率は $\eta_\theta = 7.0 \times 10^{-1}$，パターン 1 の構造パラメータ θ の学習率は $\eta_\theta = 7.0 \times 10^{-4}$，パターン 2 の構造パラメータ θ の学習率は $\eta_\theta = 7.0 \times 10^{-2}$ とした．

図 4.5 は各反復ごとの MSE の値を表している．図 4.1 と違い初期値が ISTA と同じ構造になっているため，反復途中の MSE の値のブレが少ないことがわかる．また AS-ISTA と BAS-ISTA(1) は収束速度が速くなっている代わりに精度は少し悪くなっている．これは初期値が ISTA と同じ構造になっているため構造パラメータの変化が少ないためであると考えられる．

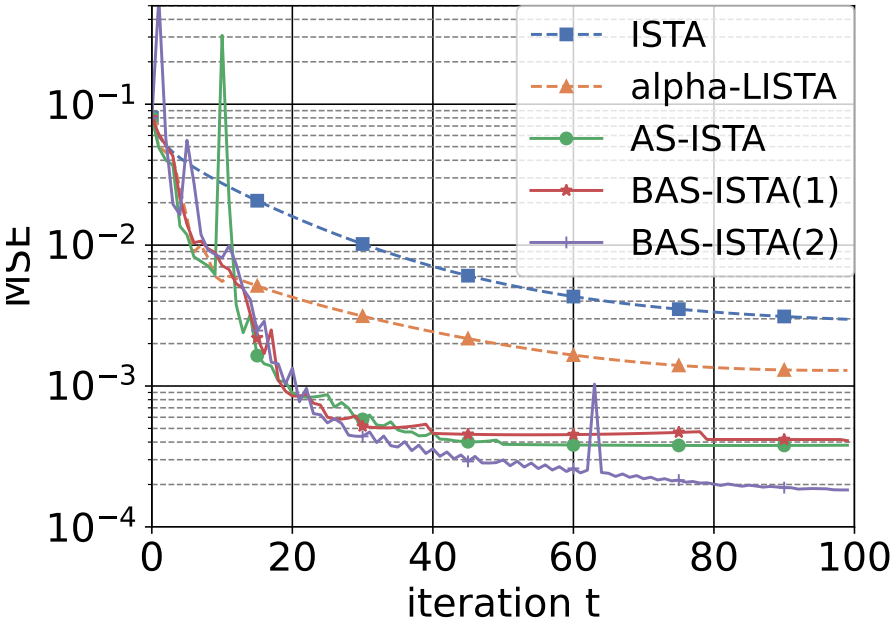


図 4.5: 各反復ごとの MSE の値

図 4.6 は各反復ごとの目的関数の値を表している．図 4.2 と同じように反復開始時は目的関数の値が大きくなっているが，最終的には同じような結果を得た．20 反復目を超えると 5 つともほぼ同じ値をとるようになった．最終的な目的関数の大きさの順番は図 4.2 と同じである．

図 4.7 は各反復ごとのステップサイズパラメータの値を表している．図 4.3 と同じように反復回数が増えるにつれてステップサイズパラメータの値が小さくなる傾向が見て取れる．

図 4.8a，図 4.8b，図 4.8c は AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) の構造パラメータの結果から，それぞれのアルゴリズムが最終的にとっている構造を可視化したものである．AS-ISTA の構造パラメータはほぼ 1.0 または 0.0 に近い値をとっている．AS-ISTA と BAS-ISTA(1) は縮小ステップが勾配降下ステップに置き換わっていることが分かる．BAS-ISTA(2) は勾配降下ステップが縮小ステップに置き換わっている部分が多いが勾配降下ステップが縮小ステップに置き換わっている部分もある．

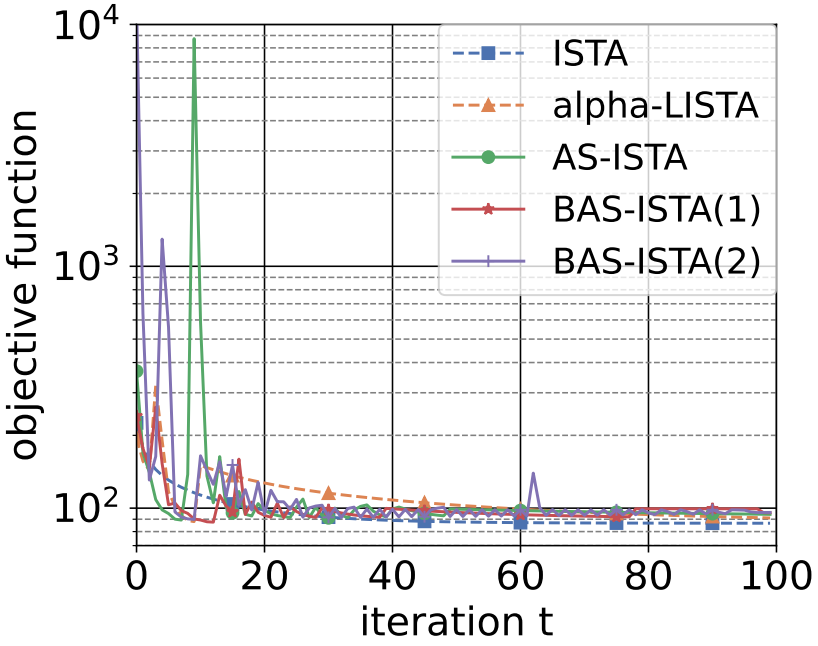


図 4.6: 各反復ごとの目的関数の値

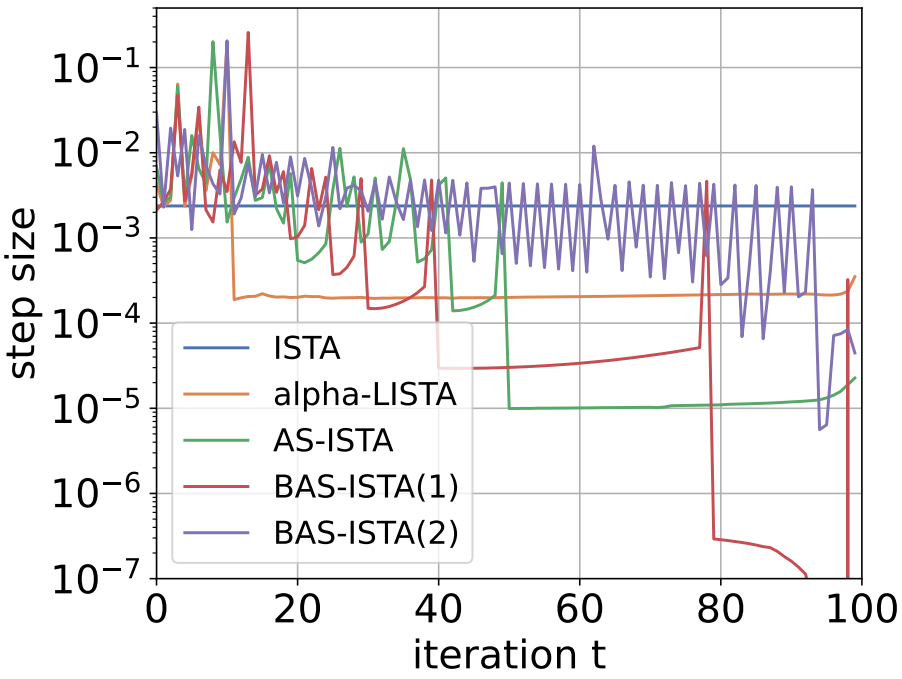
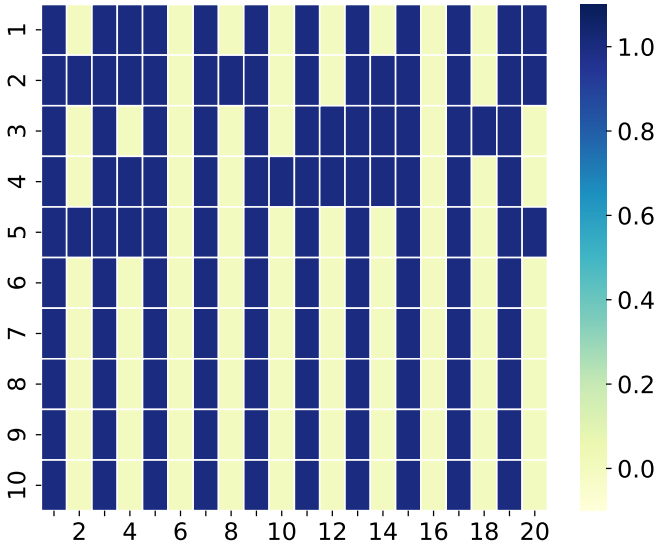
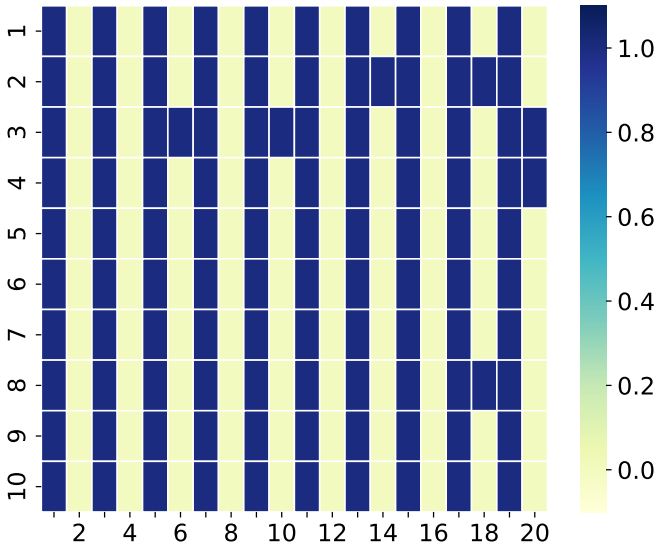


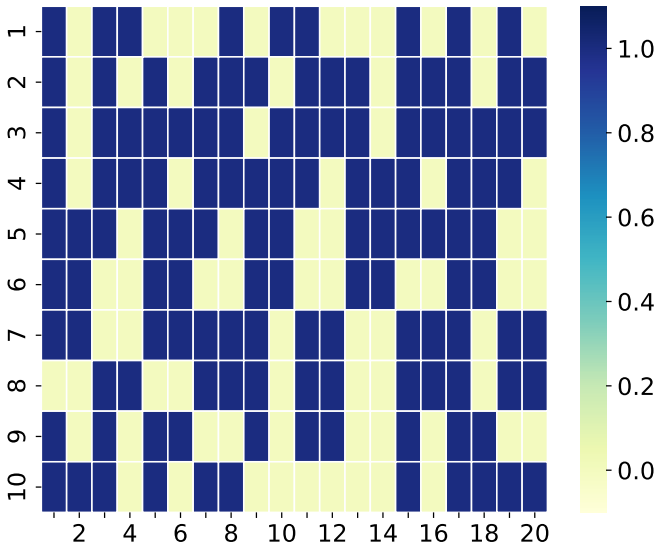
図 4.7: 反復ごとのステップサイズの値



(a) 学習後の AS-ISTA の構造



(b) 学習後の BAS-ISTA(1) の構造



(c) 学習後の BAS-ISTA(2) の構造

図 4.8: 初期値パターン 2 の構造の比較

表 4.2: 各アルゴリズムの 100 反復にかかる計算時間

アルゴリズム	計算時間（秒）	AS-ISTA 比 (%)
ISTA	1.72×10^{-2}	51.2
AS-ISTA	3.36×10^{-2}	-
BAS-ISTA(1)	1.74×10^{-2}	51.8
BAS-ISTA(2)	1.78×10^{-2}	53.0

最後に ISTA，AS-ISTA，BAS-ISTA(1)，BAS-ISTA(2) のそれぞれの 100 反復時にかかる計算時間を表 4.2 に示す．計算時間の計測は 100 回の反復を 10000 回行い，その平均値を計算した．それぞれ計算時間は，1.72×10^{-2} 秒，3.36×10^{-2} 秒，1.74×10^{-2} 秒，1.78×10^{-2} 秒であった．これより BAS-ISTA(1) は AS-ISTA に比べて約 51.8 パーセント，BAS-ISTA(2) は AS-ISTA に比べて約 53.0 パーセントの計算時間となった．

第5章　結論

本研究では，圧縮センシングアルゴリズムの ISTA に対してパラメータと構造の両方を学習させる AS-ISTA の重みパラメータの二値化を行った．計算機シミュレーションの結果，重みの二値化を行いながら学習をさせた場合でも AS-ISTA と近い精度の結果を得ることができた．また，重みを二値化したことで反復アルゴリズムの実行時間を短縮することができた．今後の課題としては，収束速度の向上や反復途中の精度の向上が挙げられる．

謝辞

本研究の全過程を通して直接の丁寧なご指導，ご助言を賜りました東京農工大学大学院工学府知能情報システム工学専攻，早川諒准教授に深く感謝いたします．本研究に対してご教授受け賜りました大阪大学大学院基礎工学研究科システム創成専攻，飯國洋二教授に深く御礼申し上げます．本研究に関して多くのご助言を賜りましたシステム創成専攻，下倉良太准教授に深く感謝いたします．本研究に関して多くのご助言を賜りました大阪大学大学院基礎工学研究科システム創成専攻，修士課程１年長久紘士さんに深く感謝いたします．最後になりましたが飯國研究室の大学院生の方々，学部生の方々に心から感謝いたします．

参考文献

[1] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[3] G. B. Passty, "Ergodic convergence to a zero of the sum of monotone operators in hilbert space," *Journal of Mathematical Analysis and Applications*, vol. 72, no. 2, pp. 383–390, 1979.

[4] P. Tseng, "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM Journal on Control and Optimization*, vol. 29, no. 1, pp. 119–138, 1991.

[5] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[6] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th international conference on international conference on machine learning*, 2010, pp. 399–406.

[7] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[8] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[9] 長久紘士, 早川諒, and 飯國洋二, "圧縮センシングアルゴリズムの構造学習," **第３８回信号処理シンポジウム**, 2023.

[10] 和田山正 and 高邉賢史, "深層展開に基づく信号処理アルゴリズムの設計—収束加速とその理論的解釈—," **電子情報通信学会 基礎・境界ソサイエティ** *Fundamentals Review*, vol. 14, no. 1, pp. 60–72, 2020.

[11] D. Ito, S. Takabe, and T. Wadayama, "Trainable ista for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, 2019.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.