北海道大学大学院博士後期課程

情報科学研究科博士論文

ニューラルネットワークの効果的な訓練のための探索と収束の制御

高瀬　朝海
(情報理工学専攻)

2018年6月

概要

近年，人工知能の分野における様々な問題に対する機械学習手法として，ニューラルネットワークが広く利用されている．ニューラルネットワークの学習では，学習の序盤において損失関数上の誤差の大きい局所解に捕捉されないように重みと呼ばれるパラメータを反復的に更新する必要がある．重みの更新手法として，勾配降下法やその発展形である AdaGrad や Adam が一般的に用いられているが，それらの方法では局所解からの脱出が十分に行われないなどの問題点がある．

本論文ではこの問題を解決することを目的として，誤差の大きい局所解への収束を避け，誤差の小さい解に収束するための効果的な学習戦略として，学習の前半は広い範囲を探索し，後半は安定した収束を行うという戦略を考えた．この戦略をニューラルネットワーク学習に適用するために，本論文では学習率とバッチサイズという 2 つのハイパーパラメータに着目し，それらが解の探索に与える効果を議論し，この戦略に基づき，ニューラルネットワーク学習のための 2 つの手法として適応的学習率調整法（ALR 法）およびバッチサイズ増加法を提案した．ALR 法は，学習中の訓練誤差ができるだけ小さくなるようにエポックごとに動的に学習率を制御する手法であり，バッチサイズ増加法は学習中のバッチサイズを単調に増加させる手法である．この分野のベンチマークとして知られる MNIST および UCI 機械学習リポジトリ (Car Evaluation, Wine, Letter Recognition) データセットを用いた実験により，提案手法の有効性を実証した．

目 次

第1章　序論

1.1　研究目的

ニューラルネットワークは，1940〜1950 年代以降に，人工知能研究の一部として Minsky やその他多くの研究者らによって活発に研究されてきたシステムであり，図 1.1 のように，生物の神経回路網に模して，ニューロンを変数に対応させ，それらを重み付きのリンクによって結合することにより，変数間の関係を表現する．特に，階層的なネットワークは，数学的には，重み（ベクトル $\boldsymbol{w}$）をパラメータとし，入力（ベクトル $\boldsymbol{x}$）と出力（ベクトル $\boldsymbol{y}$) を関連付ける関数 $\boldsymbol{y} = f_{\boldsymbol{w}}(\boldsymbol{x})$ を表現する．

重み $\boldsymbol{w}$ の値を適切に設定するためには，いわゆる「学習」という処理を行う必要がある．典型的には，訓練データと呼ばれる望ましい入出力関係を満たす $\boldsymbol{x}$ と $\boldsymbol{y}$ の多数の対を用意し，そのデータにできるだけ良く適合（フィット）するように $\boldsymbol{w}$ の値を求める．そのアルゴリズムの基本的な考え方は，望ましい入出力関係と実際の入出力関係の違いを誤差関数（あるいは損失関数）と呼ばれる関数 $E(\boldsymbol{w})$ として定義し，その値を最小化するような $\boldsymbol{w}$ を探索することである．

実用上の観点から，ほとんどの場合，$E(\boldsymbol{w})$ を微分可能な関数として定義し，探

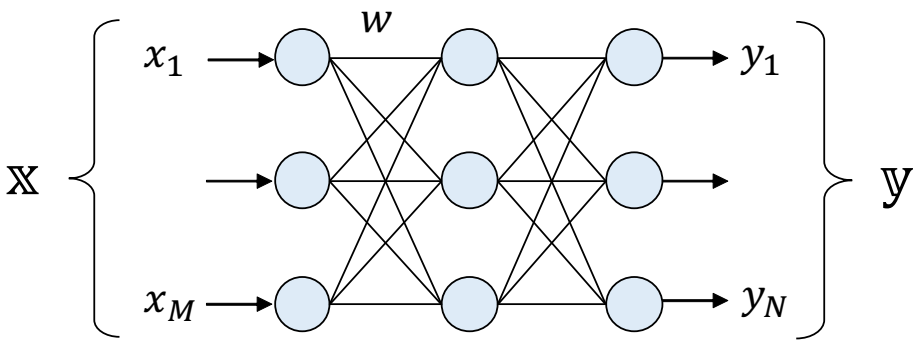


図 1.1: ニューラルネットワークの説明図.

索は最急勾配降下法，すなわち $\boldsymbol{w}$ の適当な初期値から開始して，$E(\boldsymbol{w})$ の勾配がもっとも減少する方向に $\boldsymbol{w}$ の値を適当なステップ幅で反復的に更新していく手法を基本とする．この方法は近年，大きく発展し，その振る舞いに確率的な変化を持たせる確率的勾配降下法（SGD 法）や，ステップの方向や幅を精細に制御できる AdaGrad(Duchi, Hazen, & Singer, 2011) や Adam(Kingma & Ba, 2015) などが開発されてきた．

このような探索技術の発展と相まって，2010 年代になるとニューラルネットワークの階層の数を多くしたシステムの実用性が向上し，いわゆる深層学習（ディープラーニング）と呼ばれる分野の研究が盛んとなってきた．また，それに合わせて，探索技術のさらなる発展も期待されている．

本研究では，ニューラルネットワークの学習に用いられる探索技術をさらに発展させることを目的として，学習過程をおよそその前半と後半とに分けて，それぞれにおいてアルゴリズムが（後に述べる意味で）効果的に振る舞い，最終的に望ましい解に安定して収束するように制御を行う方法について議論する．

誤差の大きい局所解から抜け出し，精度を向上させるためには，学習の前半は誤差の大きい局所解を避けて広い範囲を探索し，後半は誤差の小さい解に向けて安定した収束を行うことが重要であると考えられる．そこで筆者は，学習の前半は広く探索し，後半は安定した収束を行うという学習戦略が効果的であると考えた．

この戦略に関係する一般的な最適化問題の解法として，シミュレーテッド・アニーリングがある．これは，関数値が増加する方向にも確率的にパラメータを動かすことで広く探索を行い，その確率を徐々に小さくすることで安定した収束を図るという狙いをもつ．しかし，シミュレーテッド・アニーリングは勾配を求めることのできない誤差関数において適用できる汎用的な手法であるため，勾配情報を利用できるニューラルネットワークにおいては一般的に用いられない．

この戦略をニューラルネットワーク学習に適用するために，著者はニューラルネットワークのハイパーパラメータの一部である学習率（パラメータの更新ステップの幅を決定する値）およびミニバッチ（パラメータ更新を一括で行う訓練データの集合）のサイズに着目した．学習中，学習率は固定するか減少させ，バッチサイズは固定して用いるのが一般的であるが，本研究では，学習率およびバッチサイズを増減させることで，前述の学習戦略に従った効果的な探索と収束を実現する制御手法を提案する．

1.2　本論文の構成

本論文では，上記の内容を次の構成で述べている．

第 1 章では，研究目的および本論文の構成について述べている．

第 2 章では, ニューラルネットワークおよびその学習について説明している. まず代表的なニューラルネットワークである多層パーセプトロンや畳み込みニューラルネットワークを紹介する. その後,SGD 法や AdaGrad 法などのパラメータ更新手法を紹介し, 本論文の提案手法に関係するミニバッチ学習や学習率についても説明を加え, さらに非凸性を考慮した学習戦略について述べている.

第 3 章では，学習率の動的変化と誤差の小さな解への収束との関係，および提案手法である適応的学習率調整法（ALR 法）について説明している．まず，学習率決定に関する先行研究を紹介した後，訓練誤差最小化に基づく ALR 法を説明するとともに，本手法を用いた学習アルゴリズムおよび 3 つの主要パラメータの影響について説明している．その後，ALR 法を用いた実験結果に基づき，手法の有効性について議論している．ALR 法と一般的なニューラルネットワークの更新手法である SGD 法を用いた場合とで，訓練誤差およびテスト精度の結果を示し，ALR 法が汎化性能を向上させ得ることを示している．また，学習中の学習率の推移から，探索と収束の制御に関する本手法の効果について考察を行っている．さらに，更新手法の比較や，主要パラメータの影響，畳み込みニューラルネットワークを用いた結果，複数のデータセットを用いた評価結果などについても，実験結果に基づき考察を行っている．最後に ALR 法の問題点と今後の課題について述べている．

第 4 章では，バッチサイズの動的増加とフラットな局所解への収束との関係，および提案手法であるバッチサイズ増加法について説明している．まず，非凸最適化問題の関連研究を紹介し，大きなバッチサイズを用いたときに汎化性能が悪化するという一般的な経験則について述べている．本論文では，その理由を損失関数の非定常的な特性に基づき説明し，それを発展させ，効果的な学習を行うバッチサイズ増加法を導いている．その後，バッチサイズ増加法を用いた実験結果に基づき，手法の有効性について議論している．訓練誤差やテスト精度の結果を示し，バッチサイズ増加法がバッチサイズを固定した場合に比べて，過学習を抑制できていることを示している．また，学習の前半および後半の探索範囲を示し，バッチサイズ増加法を用いることによって探索と収束を第 2 章で述べた学習戦略に基づいて効果的に制御できていることを確認している．

第 5 章では，本論文のまとめおよび今後の課題や展望について述べている．

第2章　ニューラルネットワークおよび非凸最適化問題の学習戦略

2.1　多層パーセプトロン

ニューラルネットワークの代表的な例として，多層パーセプトロン (Multilayer perceptron: MLP) が挙げられる．MLP の構造を図 2.1 に示す．

MLP は入力層，隠れ層（中間層），出力層に分けられ，入力層に入力されたデータ $x_1, x_2, \cdots, x_I$ が，ユニット同士を結ぶ結合に付与された重みを乗じられながら，活性化関数を通り，出力層から $y_1, y_2, \cdots, y_N$ として出力される．入力層のユニット数 I は，使用するデータの特徴の数に一致するように定め，出力層のユニット数 N は，使用するデータの分類するクラス数に一致するように定める．各隠れ層のユニット数は同じである必要はなく，隠れ層の層数も自由に設定することがで

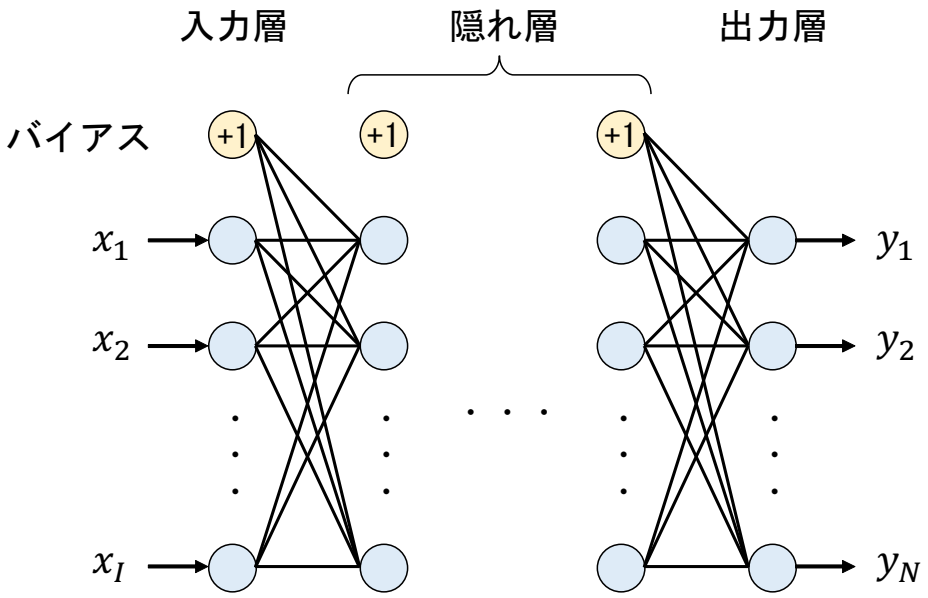


図 2.1: 多層パーセプトロンの構造.

きる．

ユニット間の結合を図 2.2 に示す．ここで，$w_{ij}^{(l)}$ は第 l 層の i 番目のユニットと第 $l+1$ 層の j 番目のユニットとの結合に付与された重みであり，$b_j^{(l)}$ は第 l 層から第 $l+1$ 層の j 番目のユニットへの入力に加えられるバイアスである．第 l 層の他のユニットからの入力も考慮すると，第 $l+1$ 層の j 番目のユニットへの入力は，以下の式によって表される．

$$u_j = \sum_i w_{ij}^{(l)} x_i + b_j^{(l)} \tag{2.1}$$

第 $l+1$ 層の j 番目のユニットからの出力は，u_j を活性化関数 f に通して，以下のように与えられる．

$$z_j = f(u_j) \tag{2.2}$$

中間層の活性化関数にはロジスティック関数や双曲線正接関数などのシグモイド関数が用いられることが多く，また近年は以下の式で与えられる，正規化線形関数（ReLU 関数）がよく用いられる．

$$f(u) = \max(u, 0) \tag{2.3}$$

本研究で行う実験では，中間層の活性化関数にはすべて ReLU 関数を用いている．

多クラス分類問題の場合，出力層 ($l = L$) の活性化関数には，以下に示すソフトマックス関数を用いる．

$$y_k = \frac{\exp(u_k^{(L)})}{\sum_{j=1}^{N} \exp(u_j^{(L)})} \tag{2.4}$$

この式で，出力値の総和が常に 1 になるように正規化が行われている．出力層からの出力は，正解データとの差を用いることで，損失関数（誤差関数）を形成する．多クラス分類問題の損失関数には，以下の式で与えられる，交差エントロピーが使用される．

$$E(\theta) = -\sum_{m=1}^{M} \sum_{n=1}^{N} d_n^{\{m\}} \log y_n(x^{\{m\}}; \theta) \tag{2.5}$$

ここで，M は訓練データ数であり，$x^{\{m\}}$ は m 番目のサンプルの入力データを，$d_n^{\{m\}}$ は m 番目のサンプルにおける正解値の n 番目の値（出力層 $l = L$ のユニット n における出力値に対応）を表す．また，θ は重みとバイアスを含んだベクトルで

あり（以下単にパラメータと呼ぶ），損失関数を勾配降下法によって最小化するように θ を更新する．勾配降下法は以下の式

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla E(\theta_t), \tag{2.6}$$

によって行われる．ただし，η は学習率と呼ばれる実数である．この式を実行するためには，誤差関数 $E(\theta)$ の $\theta = \theta_t$ における偏微分値である $\nabla E(\theta_t)$ の値を知る必要がある．しかし，ニューラルネットワークは多層で非線形な活性化関数をもつため，誤差関数の式は複雑な入れ子構造になり，直接的に偏微分を求めることが困難である．

そこで，出力層から入力層に向けて逆向きに $\nabla E(\theta_t)$ の値を求めることができる，誤差逆伝播法という手法が提案されており，ニューラルネットワーク学習に一般的に利用されている．誤差逆伝播法の結果の式を以下に示す．

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} (\theta_{kj}^{(l+1)} f'(u_j^{(l)})) \tag{2.7}$$

$$\frac{\partial E_n}{\partial \theta_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{2.8}$$

出力層 $l = L$ のユニット j における出力値を y_j，正解値を d_j とすると，$\delta_j^{(L)}$ が $y_j - d_j$ で与えられるので，これを利用して式 (2.7) で各層の $\delta_j^{(l)}$ を計算し，それによって式 (2.8) で $\partial E_n / \partial \theta_{ji}^{(l)}$ の値を求めることが可能になる．得られた $\partial E_n / \partial \theta_{ji}^{(l)}$

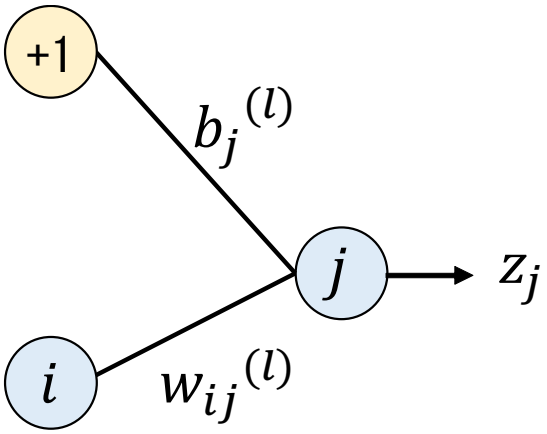


図 2.2: ユニット間の結合.

は，式 (2.6) に利用され，パラメータの更新が行われる．

2.2　畳み込みニューラルネットワーク

画像認識タスクでは，通常各ピクセルの値をニューラルネットワークの入力層の各ユニットに入力するが，近傍のピクセルとの関係，すなわち入力の順序も重要になる．そのため，前節で紹介した多層パーセプトロンではなく，畳み込みニューラルネットワークを用いることが多い．

これは大きく分けて，畳み込み層，プーリング層，全結合層から成る．一般的な畳み込みニューラルネットワークの構成は，畳み込み層とプーリング層のペアを複数回繰り返した後，最後に全結合層を挿入するというものである．全結合層は 2 層や 3 層程度のものが利用されるが，それは前節で示した多層パーセプトロンと同じ性質である．ここでは，畳み込みニューラルネットワークに特有の，畳み込み層およびプーリング層について説明を加える．

畳み込み層では，画像にフィルタを畳み込むことにより，フィルタに応じた特徴を画像から抽出する．単純に畳み込みを行うと，画像のサイズが小さくなるが，元の入力データに縁をつけてから畳み込みを行うパディングによって，画像サイズを畳み込み後も維持することができる．また，フィルタを 1 ピクセルずつ動かすのが基本であるが，ストライドを設定することで，複数のピクセルずつ動かすことも可能である．

畳み込み層で行われる具体的な計算を以下に示す．

$$u_{ijm} = \sum_{k=0}^{K-1} \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqkm} + b_{ijm} \tag{2.9}$$

ここで，K はチャネル数，H は正方形の画像の 1 辺のピクセル数を表す．また，$z_{i+p,j+q,k}^{(l-1)}$ は，直前の $l-1$ 層から受け取った画像の，k チャネルの $(i+p, j+q)$ のピクセルの値であり，$h_{pqkm} + b_{ijm}$ はそれに対応する m 番目のフィルタであり，b_{ijm} はバイアスである．

プーリング層は，畳み込み層の後に挿入されることが多く，画像内の対象の位置感度を低下させることで，対象の位置がずれた場合にも出力が変化しないようにする．画像を複数のプーリング領域に分割し，各領域においてプーリングを実行する．それぞれのプーリング領域は一部が重なるように設定することもでき，そ

の重なり具合は，畳み込みと同様，ストライドを設定することで任意に決めることができる．

プーリングの種類として，最大プーリング，平均プーリング，Lp プーリングが一般的である．最大プーリングは，その領域のピクセル全てを，各プーリング領域に含まれるピクセルの最大値をもつ 1 つのピクセルに置き換える方法であり，平均プーリングは各プーリング領域に含まれるピクセルの平均値をもつ 1 つのピクセルに置き換える方法である．Lp プーリングは，最大プーリングと平均プーリングを両方表すことのできる，一般性をもった表記である．本研究の畳み込みニューラルネットワークを用いた実験においては，最大プーリングを用いているので，最大プーリングの計算を以下に示す．

$$u_{ijk} = \max_{(p,q)\in P_{ij}} z_{pqk} \tag{2.10}$$

ここで，u_{ijk} は，プーリング後の，チャネル k の (i,j) におけるピクセルの値であり，z_{pqk} はプーリング前のピクセルの値を，P_{ij} は各領域のピクセルの集合を表す．

2.3　勾配降下法

複数の凹凸を含む非凸な関数において，大域解を解析的に求めることは困難である．そのため，数値計算によって局所解を求める方法が有効であり，勾配降下法（最急降下法）を利用することで，局所解を探索することが可能になる．

勾配降下法のアルゴリズムは

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla E(\theta_t), \tag{2.11}$$

によって表され，図 2.3 に示されるように，勾配の逆方向に勾配の大きさに応じてパラメータの更新が行われる．ここで θ_t は t 回目の更新におけるパラメータであり，E は最小化したい目的関数（ニューラルネットワークでは損失関数），η は学習率である．学習率は更新のステップ幅を決定する．勾配降下法は誤差が小さくなるようにパラメータを更新することができるが，一旦局所解にとらわれたら，抜け出すことができないため，大きな目的関数値をもつ局所解に収束する可能性もある．2.6 節で述べるシミュレーテッド・アニーリングは，更新にランダム性を加えることで，この欠点を緩和している．

ニューラルネットワークの学習に勾配降下法を適用したものはバッチ勾配降下法と呼ばれる．そこでは，損失関数がすべての訓練データの誤差を用いて決定される．すべての訓練データを利用することで，損失関数の形状が固定され，パラメータの更新が行いやすくなる．しかし，すべてのデータをメモリ上に記憶する必要があり，多くの記憶容量を必要とする．そこで，訓練データの中から，ランダムに 1 つのサンプルを取り出して，損失関数を決定し，更新を行う確率的勾配降下法（Stochastic Gradient Descent: SGD）が，ニューラルネットワークの更新手法として用いられることがある．この手法は記憶容量を多く必要としない反面，計算時間が多くかかるという問題もあるため，実際には，複数の訓練サンプルの集合であるミニバッチ単位で更新を行う，ミニバッチ確率的勾配降下法が一般的に利用されている．どのサンプルをミニバッチに含めるかをランダムに決定するため，これもまた確率的勾配降下法の一種であり，SGD と呼ばれている．ミニバッチに含めるサンプル数（バッチサイズ）は計算時間に影響を与えるが，それだけではなく汎化性能にも影響を与えるということを 4 章で議論する．

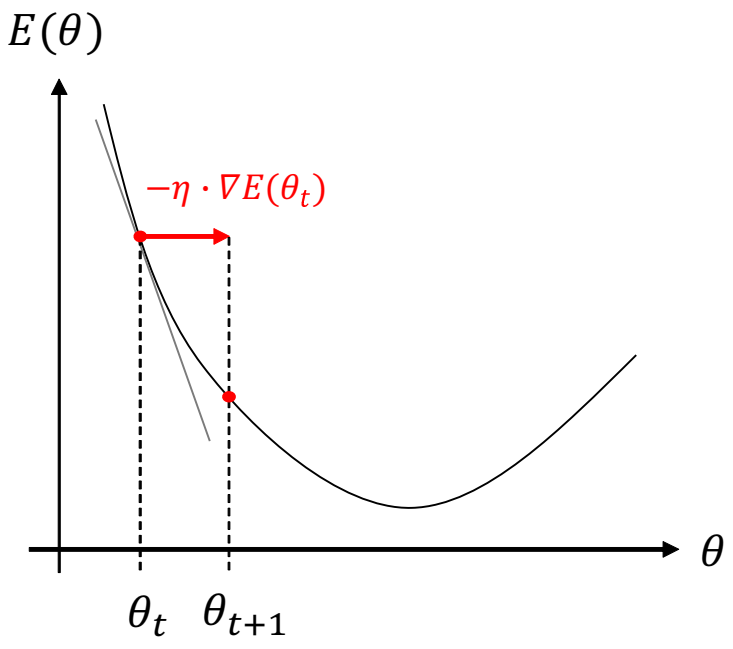


図 2.3: 勾配降下法.

2.4　ニューラルネットワークのオプティマイザ

確率的勾配降下法はニューラルネットワークのオプティマイザ（最適化手法）として広く利用されているが，前節で述べたように，多数の凹凸を含む損失関数上では，SGD により更新されたパラメータは局所解から抜け出すことができないため，望ましくない局所解に収束することも起こり得る．そこで，SGD を改良したオプティマイザも数多く提案されており，状況に応じて適切なオプティマイザが利用されている．AdaGrad(Duchi, Hazen, & Singer, 2011), RMSprop(Tieleman & Hinton, 2012), AdaDelta(Zeiler, 2012), Adam(Kingma & Ba, 2015) などが有名である．

特に広く利用されているオプティマイザの例として，AdaGrad について以下に述べる．AdaGrad では以下の式によって重みが更新される．

$$r_t = r_{t-1} + \left(\frac{\partial E}{\partial \theta_t}\right)^2 \tag{2.12}$$

$$\eta_t = \frac{\eta_0}{\sqrt{r_t} + \varepsilon} \tag{2.13}$$

$$\theta_{t+1} = \theta_t - \eta_t \nabla E(\theta_t) \tag{2.14}$$

ここで式 (2.12) の θ_t は t 回目の更新前における各パラメータを表しており，パラメータごとに学習率 η_t を求める．$t(\geq 1)$ は更新の回数を，$\nabla E(\theta_t)$ は t 回目の更新時の損失関数 $E(\theta_t)$ の勾配を，η_t は t 回目の更新における学習率を，η_0 は初期学習率を，ε は発散を避けるための小さな定数をそれぞれ表す．また，$r_0 = 0$ である．重みは式 (2.14) を用いることによって更新され，学習率は式 (2.12) および (2.13) を用いて計算される．

2.5　非凸性を考慮した学習戦略

機械学習の目的は，訓練データにモデルをフィットさせることではなく，汎化性能を向上させることではあるが，汎化性能を向上させるためには，オーバーフィッティングとならない程度に低い訓練誤差を達成することが重要である．通常，損失関数は複数の凹凸を含むため，大域解に収束することは難しいが，誤差の大きい局所解によるトラップを避け，誤差の小さい局所解への収束を行うことが重要である．前節で述べた Adam などのオプティマイザは，SGD に比べ，鞍点と呼ば

れる，ある次元からは山の形状で別の次元からは谷の形状になっている部分から抜け出しやすくなるといった利点があり，より低い誤差をもつ局所解への収束が行いやすくなっている．訓練データとテストデータは，それぞれの分布が大きくは異ならないため，訓練データにおいて小さい誤差を達成することは，汎化性能の向上につながるのは確かである．

しかし，汎化性能に影響を与えるのは誤差の大小だけではないということを Keskar ら (Keskar et al., 2017) は指摘した．彼らはシャープな局所解およびフラットな局所解というものを仮定し，シャープな局所解にパラメータがとらわれるならば，テスト精度が悪化する傾向にあると述べた．シャープな局所解に対する理論的な裏付けはなかったが，多くの実験的な結果から仮説を立てた．彼らの研究は本論文の動機づけとなったので，以下に詳細な説明を与える．

図 2.4 に示されるように，シャープな局所解とフラットな局所解が学習時の関数において定義されるとき，テスト時の関数における誤差の値は，フラットな局所解よりもシャープな局所解においてより大きくなる傾向にある．これは，学習時とテスト時で入力データが異なるため，損失関数の形状がわずかに変化し，シャープな局所解ではフラットな局所解よりもその影響を受けやすいからである．したがって，シャープな局所解にパラメータがとらわれるならば，汎化性能が悪化する傾向にある．この議論は現時点ではまだ精密性に欠けるとはいえ，そのアイデアにはある程度の説得力が感じられる．

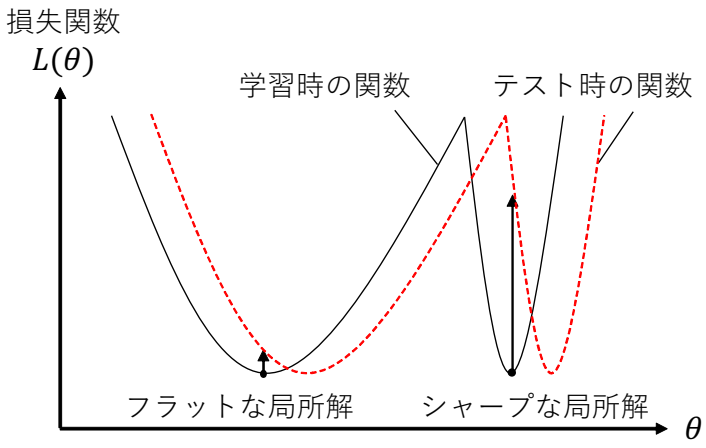


図 2.4: フラットな局所解とシャープな局所解の汎化性能の違い (Keskar ら (Keskar et al., 2017) に基づく). θ 軸は変数を表し，$L(\theta)$ 軸は損失関数の値を表す．

Keskar らの議論に基づいて汎化性能を向上させるとすれば，(1) できるだけ誤差の小さい値に収束すること，(2) シャープな局所解を避け，フラットな局所解に収束すること，のいずれもが重要である．本研究では，それらを達成するために，ニューラルネットワークのハイパーパラメータである学習率とバッチサイズに着目し，学習率と (1) との関係を 3 章に示す研究で議論し，バッチサイズと (2) との関係を 4 章に示す研究で議論した．

まず，上述の (1) および (2) を達成するための，効果的な学習戦略を考えた．それは，図 2.5 に示すように，学習の前半は広く探索を行い，学習の後半は近傍の解に向けて安定した収束を行うというものである．これにより，誤差の大きい解やシャープな解を避けつつ，誤差の小さい解やフラットな解に収束することが期待される．本研究では，学習率およびバッチサイズを学習中に変化させることが，この学習戦略とどう結びつくのかを議論し，学習戦略に基づいた効果的な学習手法を提案する．

2.6　シミュレーテッドアニーリング

シミュレーテッド・アニーリングは，前節で述べた学習戦略に基づいた乱択アルゴリズムであり，一般的な最適化問題において，局所解を避けつつ探索を行う手法として広く知られている．

勾配降下法において，探索は常に関数値が減少する方向に進むが，シミュレーテッドアニーリング（アルゴリズム 1）においては，温度パラメータによって制御

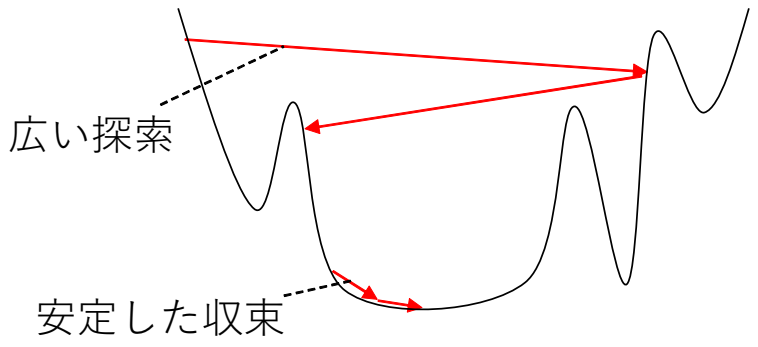


図 2.5: 非凸最適化問題における効果的な学習戦略. 学習の前半は広く探索し，後半は安定した収束を狙う．

Algorithm 1 シミュレーテッド・アニーリング

Require: 全更新回数: S
　　初期パラメータ: x_0
　for t from 0 to $S-1$ **do**
　　Choose $x' \leftarrow$ a random element from $\mathcal{N}(x_t)$
　　if $f(x') \leq f(x_t)$ **then**
　　$x_{t+1} \leftarrow x'$
　　else
　　$x_{t+1} \leftarrow x'$ with probability $e^{-[f(x')-f(x_t)]/\mathrm{Temp}_t}$
　　otherwise $x_{t+1} \leftarrow x_t$
　　end if
　end for
　return x_S

された確率に基づいて，探索は関数値が増加する方向にも進む．アルゴリズム 1 において，x_t はステップ t でのパラメータを，$\mathcal{N}(x_t)$ は x_t の近傍にある探索点の集合を，Temp_t はステップ t での温度を示す．温度は通常探索中徐々に減少させるので，関数値が増加する方向に探索が進む確率もまた減少する．もし温度減少の範囲が無限小ならば，パラメータは常に大域解に収束する (Lundy & Mees, 1986). 実際の使用においては，温度の変化は適切にスケジューリングされる必要がある．

シミュレーテッド・アニーリングは勾配の計算できない関数においても利用可能な汎用的な手法であるため，勾配の計算できるニューラルネットワークでは実用的に使用されない．

また，関連手法である量子アニーリングは主に組み合わせ最適化問題を解くのに用いられる．量子アニーリングを組み込んだ最初の商業コンピュータである D-Wave マシン (Johnson et al., 2011) が，近年注目を集めている．

第3章　訓練誤差に基づいた適応的学習率調整法

3.1　導入

ディープラーニングは近年様々な画像分類タスク (Krizhevsky, Sutskever, & Hinton, 2012; Wu & Gu, 2015; Shi, Ye, & Wu, 2016) や音声認識タスク (Hinton et al., 2012; Sainath et al., 2015; Fayek, Lech, & Cavedon, 2017) において優れた性能を示している．この成功は主に勾配降下法に基づく最適化アルゴリズムで用いられるパラメータ更新手法の改良によるものである．AdaGrad(Duchi, Hazen, & Singer, 2011) や，RMSprop(Tieleman & Hinton, 2012)，AdaDelta(Zeiler, 2012)，Adam(Kingma & Ba, 2015) などの SGD に基づいた更新手法が現在広く用いられており，タスクごとに適した更新手法を選択することでより良い性能につながる．

これらの更新手法を用いる際，ユーザは初期パラメータを設定する必要がある．特に初期学習率は重要であり，不適切な学習率を用いるならば，誤差の値が他の局所解と比べて小さくないような望ましくない局所解にモデルのパラメータがとらわれてしまう．したがって，これらの更新手法の主な欠点は適切に調整するのが困難な，センシティブなパラメータを有することであるといえる．

このようなセンシティブなパラメータを用いない手法として，Daniel ら (Daniel et al., 2016) によるステップサイズ制御手法が挙げられる．これは，学習率による更新のステップサイズを，その初期設定に依存しない強化学習 (Sutton & Barto, 1998) によって自動的に決定する手法である．別の手法として，Kanada(Kanada, 2016) による LOG-BP アルゴリズムが挙げられる．これは，誤差逆伝播法を遺伝的アルゴリズムと結びつけることで学習率を指数関数的に減少させる手法である．

学習率を調整する簡単なアプローチは，例えば 100 エポックといった，ある決まった学習エポックごとに，0.1 のような定数を学習率に乗ずることである．このアプローチはテスト精度を改善させるために実用的に用いられている．しかし，こ

のアプローチは次の乗算時までずっと学習率が固定されるという点で柔軟性に欠ける．

本論文では，学習中，訓練誤差を最小化するために木探索を利用して値を増加もしくは減少させることで，学習率を動的かつ自動的に調整するという，より柔軟性のある手法を提案する．上述の簡単なアプローチとは異なり，本手法ではエポックごとに複数の学習率を用いた学習を独立に並行して行い，最も小さな訓練誤差（すべての訓練サンプルの交差エントロピー誤差の和）が得られた学習率を実際の学習率として採用する．これは，学習率の最適化プロセスとみなされるが，動的計画法や強化学習を私たちのタスクに利用することは，1) 訓練は一方向，2) 訓練誤差は解析的に計算できない，3) 各エポックの学習に時間がかかりすぎる，という理由によって不適切と考えられる．この問題を解決するために，本研究では幅優先ビームサーチに基づいた効率的な探索アルゴリズムを開発した．以下では，本手法を適応的学習率調整法あるいは ALR 法 (Adaptable Learning Rate Tree アルゴリズム) と呼ぶ．

3.2 節では，様々なパラメータ更新手法における学習率の振る舞いについて分析する．3.3 節では，ALR 法を説明する．3.4 節では，提案手法と従来手法の両方を用いた実験の結果を報告する．3.5 節および 3.6 節では，ALR 法の 3 つの主要パラメータの効果について論じる．4.7 節では，研究の要約および今後の課題について述べる．

3.2　学習率

SGD は多くの更新手法の基盤となっているので，ここではまず SGD に基づいた更新手法を説明する．それは SGD に関連した多くの研究 (Neyshabur, Salakhutdinov; Breuel, 2015; Mandt, Hoffman, & Blei, 2016; Hardt, Recht, & Singer, 2015) に共通したものである．

SGD において，パラメータの更新は以下のようにサンプルごともしくはミニバッチごとに行われる:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta E(\theta_t; x^{(i)}; y^{(i)}), \tag{3.1}$$

ここで θ はニューラルネットワークのパラメータである重み（およびバイアス）で，$x^{(i)}$ は訓練サンプル i を用いた入力，$y^{(i)}$ はラベル，E は誤差関数，η は学習率で

ある．

η を増加させることで探索の範囲を広げられるが，大きすぎる η は解への収束を困難にする．この問題を解決する効率的な方法は，学習中，学習率を減少させることである．例えば，AdaGrad では式 (3.1) の η を，以下のように定義された η_t によって置き換える:

$$r_{t+1} = r_t + \left(\frac{\partial E}{\partial \theta_t}\right)^2 \tag{3.2}$$

$$\eta_{t+1} = \frac{\eta_0}{\sqrt{r_{t+1}} + \varepsilon}, \tag{3.3}$$

ここで式 (3.2) の θ_t は t 回目の更新前における各パラメータを表しており，パラメータごとに学習率 η_t を求める．また，式 (3.3) の ε は発散を避けるための小さな定数を表す．これらの式から，r_t が増加するので η_t が減少することがわかる．

Adam のような更新手法は AdaGrad に基づいており，それらの更新式は学習率が学習中減少するように設計されている．解の探索範囲は徐々に減少し，より良い解を探すことが難しくなる．対照的に，もし探索範囲を広げるように学習率を学習中増加させるならば，収束が難しくなる．学習率の減少と増加を上手く組み合わせることで，効果的な学習手法が得られる．

ALR 法では，訓練誤差が最小になるように学習率が増加または減少する．それは式 (3.1) の η が，以下のようにエポックごとに変化することを意味する:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta E(\theta_t; x^{(i)}; y^{(i)}). \tag{3.4}$$

AdaGrad とは異なり，ALR 法では計算の効率化のために，すべての重みで共通の η_t が用いられる．

ALR 法は訓練誤差に基づいて学習率を調整するが，一般に，訓練誤差の過剰な減少は学習データへの過学習につながり得る．しかし，過学習は重み上限 (Srebro & Shraibman, 2005) やドロップアウト (Srivastava et al., 2014) といった手法を用いることで抑制できる．

3.3　ALR 法

3.3.1　ALR 法

本節では，訓練誤差に基づいて学習率を調整する ALR 法について述べる．3.1 節で述べたように，学習は各訓練エポックの間，複数の学習率を用いて独立に行われ，最も小さい訓練誤差が得られたときの学習率がそのエポックでの学習率として採用される．訓練誤差は誤差関数の値であるので，ALR 法では間接的に誤差関数の形状を利用していることになる．形状の情報を利用せずに誤差関数の最小値を理論的に見つける方法が提案されているが (Song et al., 2016)，特定の誤差関数に依存するので，応用の範囲は限られている．

ALR 法では学習中の状態遷移を，木構造を用いて実現する．エポック t からエポック $t+1$ への状態遷移を図 3.1 に示す．各ノードは学習中のエポックでの状態を表す．状態は学習率 η とパラメータ θ の対 (η, θ) であり，それが各ノードに記憶される．エポック t での親ノードは N 本の枝を有しており，それぞれの枝に異なったスケールファクター $r_1, \ldots, r_N$ が割り当てられている．もし n 番目の子ノードがエポック t で選択されたならば，エポック $t+1$ での学習率は，エポック t での学習率にスケールファクター r_n を乗ずることで得られる．エポック t から $t+1$ への遷移で選ばれたスケールファクターを r_{n_t}（$n_t \in \{1, 2, \ldots, N\}$）と表すとき，両エポックにおける状態の関係は以下の式によって与えられる:

$$\eta_{t+1} = \eta_t \cdot r_{n_t}. \tag{3.5}$$

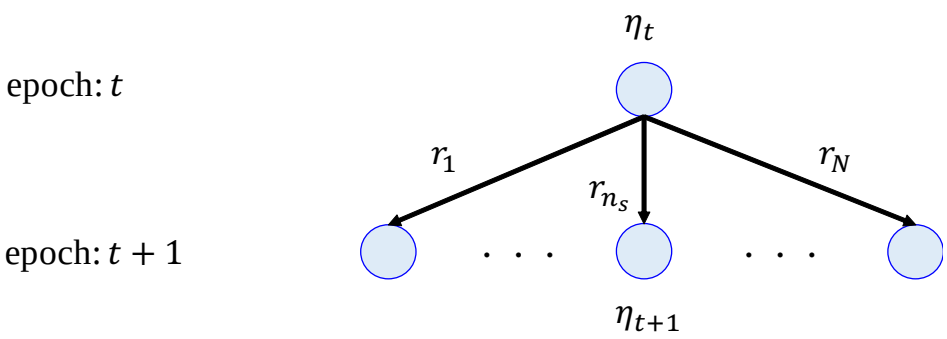


図 3.1: エポック t からエポック $t+1$ への状態遷移. N: 分岐数; r_n: 各枝のスケールファクター; η_t : エポック t での学習率.

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta E(\theta_t; x^{(i)}; y^{(i)}), \tag{3.6}$$

式 (3.5) を繰り返し用いることで，以下のように任意のエポックでの学習率が得られる:

$$\eta_s = \eta_0 \prod_{t=1}^{s-1} r_{n_t}, \tag{3.7}$$

ここで η_0 は初期学習率を表す．

実際，図 3.2 に示すように，ALR 法は多数の状態遷移を組み合わせたものであり，探索は木構造を用いて行われる．効率的に探索を行うために，ALR 法では，3.3.4 節で描写する幅優先ビームサーチを用いる．

3.3.2　パラメータ

ALR 法は，分岐数，スケールファクターの集合，ビームサイズという 3 つの主要パラメータを持っている．分岐数は各ノードで N と表し，スケールファクターのノードは $\{r_1, \ldots, r_N\}$ で表すことにする．ビームサイズ M は幅優先探索の有界幅を表す．もし，あるエポックのノード数 k が M を超えるならば，訓練誤差の大きい順に $(k - M)$ 個のノードは除去されることになる．

ユーザはこれらのパラメータを学習前に決める必要がある．ここで，分岐数とスケールファクターはすべてのノードに共通であり，ビームサイズは各エポックで共通である．これらの主要パラメータと補助的なパラメータ η_0 の効果は，それぞれ 3.5 節と 3.4.4 節において議論する．

3.3.3　手順

ALR 法の状態遷移の手順を，図 3.2 に示される木構造の例を用いて説明する．この例において，分岐数 N は 3，スケールファクターは $\{2.0, 1.0, 0.5\}$，ビームサイズ M は 4 である．各ノード内の数字は各エポックでの訓練誤差の順位（訓練誤差が小さいほど順位が上がる）を表している．各エポックでの状態遷移は以下のようになる:

エポック 1 → 2: ノード A において，1 エポックの学習が 3 つのスケールファクターを用いて独立に行われる．最も小さい訓練誤差を生み出した枝が選択され，状態はノード B に遷移する．ノード B 以外の 2 つのノードも次のステップのために保持される．

エポック 2 → 3:　保持された各ノードについて独立に 1 エポックの学習が行われ，次の遷移先の候補として 9 個のノードが生み出される．$M(=4)$ 個のノードが訓練誤差の小さい順に保持され，残りの 5 個のノードは除去される．状態は，最も小さい訓練誤差を生み出した C に遷移する．

エポック 3 → 4:　以前のエポックと同様に学習が行われ，状態は最も良いノード D に遷移する．このノードはノード C と直接結びついていないが，最も良いノード以外のノードも保持されてきたため，このような遷移も可能である．

ALR 法で用いられるアルゴリズムの全体をアルゴリズム 2 に示す．特徴的な点は複数の学習率と重みが訓練誤差の小さい順に保持されるという点である．このアルゴリズムでは学習は収束するまで続けられるとしているが，実用的には，学習の早期終了がより良い結果につながるだろう．

従来手法と比較して，ALR 法は，大きな分岐数とビームサイズが用いられるな

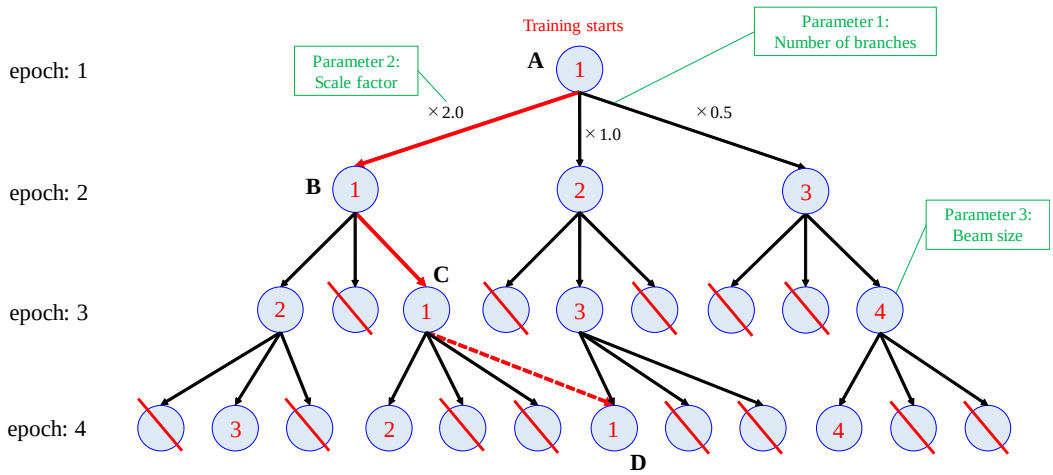


図 3.2: エポック 1 からエポック 4 までの状態遷移例. パラメータ 1:分岐数は 3. パラメータ 2:スケールファクターは 2.0, 1.0, 0.5. パラメータ 3:ビームサイズは 4. ノード上の番号は各エポックでの訓練誤差の順位を表す. 幅優先ビームサーチにおいて，番号 1 が書かれた子ノードに状態が移動する．

Algorithm 2 ALR 法

Require: ビームサイズ: M
分岐数: N
スケールファクター: $\{r_n\}_{n=1}^{M}$
学習率: $\{\eta_m\}_{m=1}^{M}$
重みとバイアス: θ, $\{\theta_m\}_{m=1}^{M}$
各エポックのバッチ数: B
Input: 訓練データ: $\{x^{(b)}\}_{b=1}^{B}$, $\{y^{(b)}\}_{b=1}^{B}$
$t \leftarrow 0$
repeat
  for $m = 1, 2, \cdots, M$ **do**
    for $n = 1, 2, \cdots, N$ **do**
      $\eta_{m,n} \leftarrow \eta_m \cdot r_n$
      $\theta_{m,n} \leftarrow \theta_m$
      for $b = 1, 2, \cdots, B$ **do**
        $E_{m,n} \leftarrow E(\theta_{m,n}; x^{(b)}; y^{(b)})$
        $\theta_{m,n} \leftarrow \theta_{m,n} - \eta_{m,n} \cdot \nabla_\theta E_{m,n}$
      end for
    end for
  end for
  $\theta \leftarrow \theta_{m_t,n_t}$ **where** $m_t, n_t \leftarrow \arg\min_{m,n}\{E\}$
  Sort $\{(\eta_{m,n}, \theta_{m,n}) | 1 \leqq m \leqq M, 1 \leqq n \leqq N\}$ in ascending order of $\{E_{m,n}\}$ and Save the first M pairs as $(\eta_1, \theta_1), \cdots, (\eta_M, \theta_M)$.
  $t \leftarrow t + 1$
until converged

らば，非常に実行時間がかかってしまう．例えば，もし分岐数が 3 でビームサイズが 20 ならば，この手法は従来手法よりも 60 倍近く時間がかかることになる．しかし，たいていの場合，ALR 法の計算時間は，比較的小さな構造をもつ多層パーセプトロン (MLP) の学習においては，大きな問題とならない．

また，再帰的ニューラルネットワークの計算コストを，並列計算を用いて減らすという Huang らの研究 (Huang et al., 2013) のように，ALR 法においても計算の並列化が効果的に計算時間を減らすことが期待される．

3.3.4　幅優先ビームサーチ

以前の節で論じたように，エポック $3 \rightarrow 4$ の遷移において，幅優先ビームサーチが，ノード C とは直接結びついていないノード D への状態遷移を可能にした．この遷移を説明する図を図 3.3 に示す．この図に示されるように，この遷移を $C \rightarrow B \rightarrow A \rightarrow \ldots \rightarrow D$ というルートによる遷移として解釈することができる．これは解を探索するという観点において，特に重要である．

重みは訓練誤差が減少するように更新される傾向にあるので，特に学習の初期において，望ましくない局所解が得られがちである．幅優先ビームサーチは望ましくない局所解を避け，より良い解を得ることを可能にする．ビームサイズの効果は 3.5.3 節で議論する．

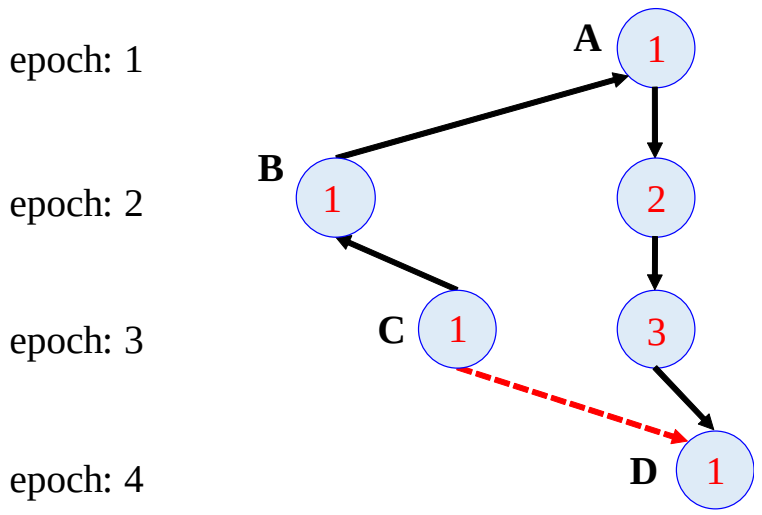


図 3.3: 幅優先ビームサーチにおける遷移の解釈に関する模式図.

表 3.1: 隠れ層 1 層をもった MLP の構造.ReLU を隠れ層に，softmax 関数を出力層に使用した.

レイヤー	ユニット数
入力層	28×28
隠れ層 ReLU	1000
出力層 softmax	10

3.4　主要な実験

3.4.1　実験条件

私たちは，MNIST データセットを用いた実験を行い，ALR 法を用いた結果と，ベースラインとしての通常の SGD を用いた結果を比較した．MNIST データセットは手書き数字画像であり，60,000 件の訓練データと 10,000 件のテストデータを含んでいる．画像サイズは 28 × 28 ピクセルで，入力値は 0 から 255 までの値をもつ各ピクセルの輝度である．各画像は 0 から 9 までの 10 クラスのいずれかに分類される．

私たちは本実験で MLP を学習させた．MLP は他のニューラルネットワークの基盤となっているため，これまで行われてきた多くの MLP に関する研究 (Egmont-Petersen et al., 1998; Park & Jo, 2016; Tang, Deng, & Huang, 2015) のように，MLP の性能を向上させることは大きな貢献になると期待される．用いた MLP の構造を表 3.1 に与える．1 層のみの隠れ層をもった MLP をモデルとして用いた．正規化線形関数 (ReLU) (Glorot, Bordes, & Bengio, 2011) を隠れ層の活性化関数として用い，ソフトマックス関数を出力層の活性化関数として用いた．用いたパラメータを表 3.2 に示す.

初期学習率を η_0，スケールファクターを r とするとき，エポック $t+1$ での学習率は $\eta_0 \cdot r^t$ と記述される．もしスケールファクターの各要素が，$a^k (a, k \in R)$ の形式で記述できるならば，取り得る学習率を特定の値 $\eta_0 \cdot a^n (n \in R)$ に制限でき，学習率推移の傾向がわかりやすくなるので，a を 2.0 に，分岐数を 3 に設定し，スケールファクターのセットとして $\{2.0, 1.0, 0.5\}$ を用いた．また，初期学習率として 0.5 を用いたので，このときの取り得る学習率の値は $0.5 \cdot 2^n$ によって与えられる．

表 3.2: 3.4.1 節の実験のパラメータ. 提案手法と従来手法とで，初期学習率，バッチサイズ，エポック数をそろえて実験を行った.

パラメータ	値
初期学習率	0.5
バッチサイズ	100
エポック数	300
分岐数	3
スケールファクター	$\{2.0, 1.0, 0.5\}$
ビームサイズ	20

実際に使用する場合，スケールファクターはこれらの値に制限される必要はない．唯一要求されることは，スケールファクターのセットが，1.0 より大きい値と 1.0 より小さい値の両方を含むということである．スケールファクターをどのように設定するのが良いかについて，3.5.2 節で述べる．

ビームサイズは 20 に設定した．ALR 法を用いた場合，それを用いない場合に比べて 1 エポック分の学習が進むまでに多くの更新を行うことになる．本実験の設定の場合，分岐数を 3 に，ビームサイズを 20 に設定しているため，ALR 法ではそれらの値を乗じた 60 倍の更新が行われる．公平な比較のため，ALR 法ではエポック数を 60 倍して結果を表示する．

ニューラルネットワークの重みの初期値として，Glorot の一様分布 (LeCun et al., 1998; Glorot & Bengio, 2010) を用いた．結果は初期値によって影響を受けるので，異なった乱数のシードを用いた初期化を行った学習を複数回行った．テスト誤識別率およびテスト誤差は毎エポック計算した．

MNIST データセットには，バリデーションデータを作るほど十分なデータがなく，それを行うとテスト精度が悪化してしまうので，バリデーションデータは用いなかった．実際の使用では，バリデーションデータを用いて，より良いパラメータを探すのがよい．

3.4.2　実験結果

図 3.4(a) と図 3.4(b) に示されるように，ALR 法は従来手法と比較して，訓練時間（エポック数）が十分長ければ，より低いテスト誤識別率と非常に小さい訓練誤差を示した．小さな訓練誤差が得られたのは，異なる学習率を用いた複数の学習の中で最も小さい誤差をもった学習が採用された結果である．

1 回の試行による学習中の学習率の推移を図 3.5 に示す．この図から，学習の前半は，学習が進むにつれて学習率が増加していることがわかる．エポック 4,380 において，その値は 65,536 という非常に大きな値に達した．テスト誤識別率が最も低い値を示したエポック 5,760 において，学習率が減少し始めた．テスト誤識別率が最も低いときに学習を止めるために，学習はバリデーション誤差だけでなく，学習率の推移を見ながら行うのがよい．

3.4.3　他の更新手法との比較

実験によって，SGD を用いた ALR 法と，SGD や, AdaGrad, RMSprop, AdaDelta, Adam を用いた従来の更新手法とを比較した．これらの手法はそれぞれ複数のパラメータをもっているが，初期学習率に対応する主要なパラメータは性能に大きな影響を与える．そこで，この実験では，そのパラメータは一般的に推奨されている値，SGD では 0.1，AdaGrad では 0.01，RMSprop では 0.001，AdaDelta では 1.0, Adam では 0.001 に設定した．他の実験条件は 3.4.1 節と同様のものを用いた．

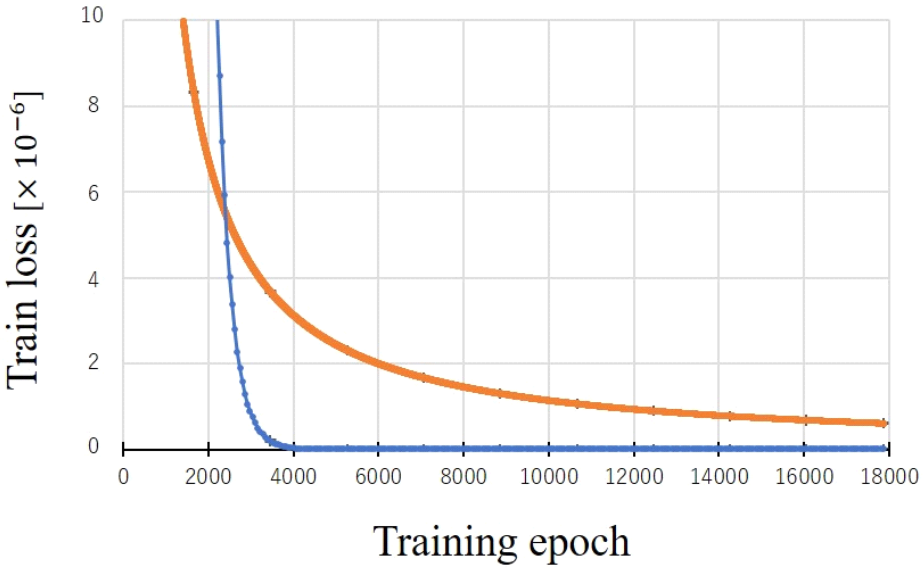
図 3.6 に示すように，ALR 法は最も低く最も安定したテスト誤識別率を示した．Adam を用いたときの最も低い値は ALR 法を用いたときとほぼ同じであったが，その推移は不安定であった．バリデーションデータを用いて早期終了を行うとしても，誤識別率が最も低いときに学習を止めることは難しいと考えられる．したがって，ALR 法が最も良い性能をもっているといえる．

3.4.4　初期学習率の影響

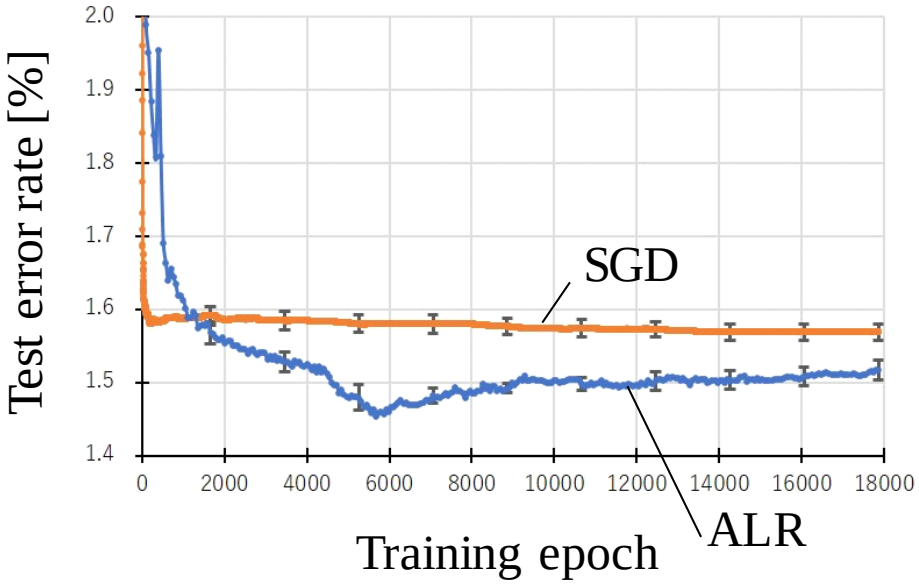
ALR 法では学習中学習率が調整されるので，従来手法よりも初期学習率の影響が小さいと考えられる．従来手法では，適切な初期学習率を見つけるために異なった値を用いてプログラムを繰り返し実行しなければならないので，パラメータチュー

ニングに非常に時間がかかってしまう．対照的に，ALR 法では順応性のある学習率を用いるので，パラメータチューニングはそれほど時間を要しない．

このことを実証するために，同じ初期学習率の SGD を用いて，ALR 法と従来手



(a) テスト誤識別率の推移.



(b) テスト誤差の推移.

図 3.4: MNIST データセットを用いた実験結果. 異なった初期重みを用いた複数の試行の平均値が示されている. エラーバーは標準誤差を表す.

法を実験によって比較した．他の実験条件は 3.4.1 節のものと同様である．

表 3.3 に示すように，ALR 法を用いたときのテスト誤識別率は，各初期学習率において，従来手法のものよりも低かった（初期学習率が 10.0 のときは両手法と

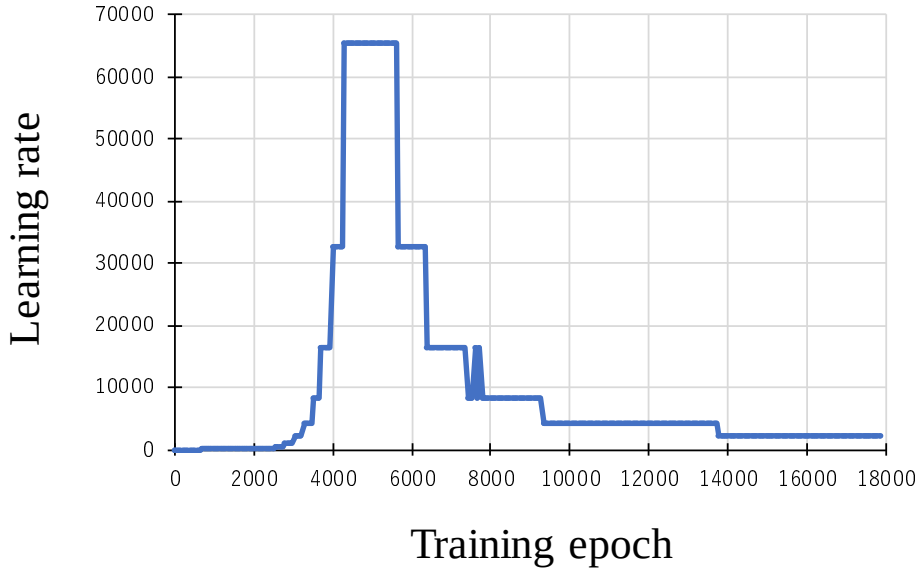


図 3.5: 初期学習率 0.5 を用いた 1 回の試行における学習中の学習率推移. 到達した最も高い学習率は 65,536 だった.

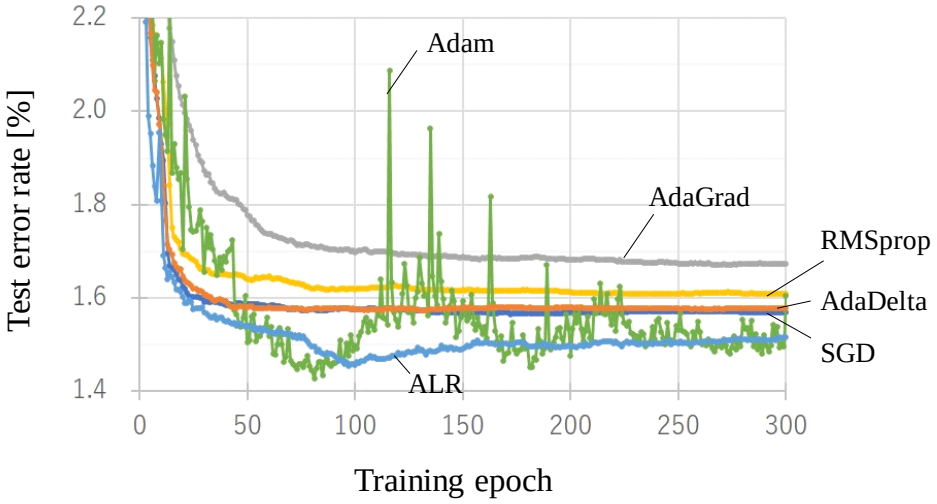


図 3.6: 他の更新手法との比較. 異なった初期重みを用いた複数回の試行の平均値が示されている.

表 3.3: 初期学習率の影響. 異なった重みの初期値を用いた複数回の試行による，最も低い誤識別率の平均を示している．重みの初期値は，両手法で同じ値を用いた．

初期学習率	テスト誤識別率 (ALR)	テスト誤識別率 (SGD)
0.01	1.54	1.80
0.05	1.50	1.70
0.1	1.50	1.69
0.5	1.43	1.57
1.0	1.43	1.53
2.0	1.46	38.78
10.0	88.65	90.24

もに学習は失敗しているが）．ALR 法を用いた場合，従来手法を用いた場合よりも，異なった初期学習率を用いたときのテスト誤識別率の変化が小さかったので，ALR 法における初期学習率の影響は，従来手法における影響より小さかったといえる．これらの手法の間のテスト誤識別率の違いは，初期学習率が 2.0 のとき特に大きかった．これらの結果は，ALR 法における初期学習率が適応的な学習率のおかげでより自由に決めることができることを示している．しかし，実用的には，0.01 から 1.0 程度の，標準の SGD で広く用いられている値を選ぶのがよい．ALR 法は悪い局所解を避けるために，学習の前半に大きくパラメータを動かす必要があるので，それらの値の中でも，0.1 と 1.0 の間から初期学習率を選ぶことが推奨される．

3.4.5　畳み込みニューラルネットワークを用いた実験

ここでは，表 3.4 に示す構造をもつ畳み込みニューラルネットワーク (CNN) を用いた実験の結果を報告する．出力層の活性化関数にソフトマックス関数を用い，それ以外の層の活性化関数に ReLU を用いた．また，各ミニバッチ内の値の平均を 0 に分散を 1 にする処理を行うバッチ正規化 (Ioffe & Szegedy, 2015) を，出力層を除く各レイヤーの出力に適用した．さらに，過学習を抑えるために，ドロップアウト法を CNN の全結合層に用いた．他の実験条件は 3.4.1 節と同様のものを用

表 3.4: 3.4.5 節で用いた CNN 構造.str. はプーリングのストライドを，p はドロップアウト法のユニット選択確率を表す.

Layer type	Channels/Units
5×5 conv, BN, ReLU	32
2×2 max pool, str. 2	32
5×5 conv, BN, ReLU	64
2×2 max pool, str. 2	64
Dropout with $p = 0.5$	64
Fully connected, BN, ReLU	1024
Dropout with $p = 0.5$	1024
Output softmax	10

いた.

図 3.7(a) は学習率の推移を示しており，図 3.7(b) はテスト誤識別率の推移を示している．MLP を用いたときの結果とは異なり，学習率は学習中に大きな値をとらなかった．学習率は基本的に減少したので，悪い局所解にとらわれたと考えられ，その結果テスト誤識別率が従来手法を用いた場合よりも悪かった.

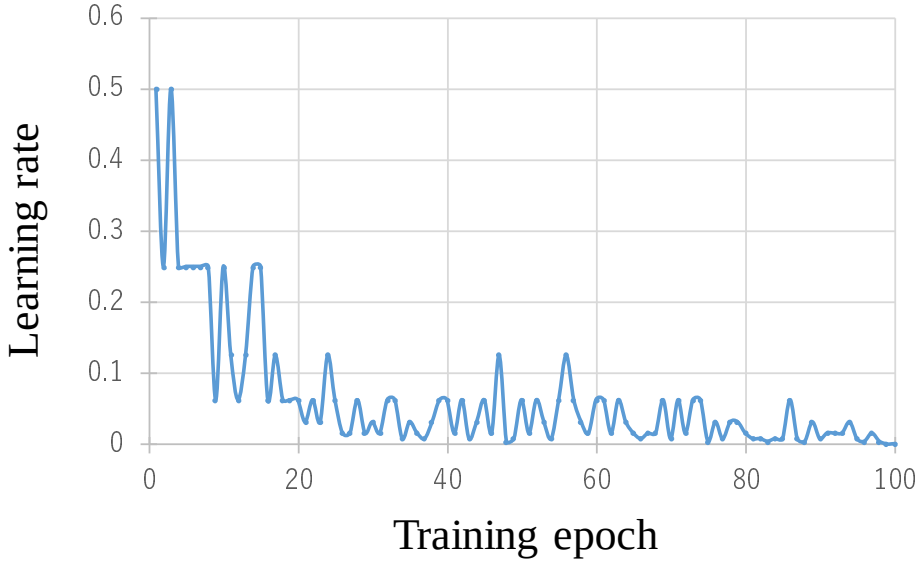
学習率が減少する傾向を示した原因は，ドロップアウト法による誤差関数の非定常性であると予想される．誤差関数が交差エントロピーであるとき，誤差関数は以下の式によって与えられる.

$$E(w) = -\sum_{c=1}^{C}\sum_{k=1}^{K} d_{ck} \log y_k(x_c; w), \tag{3.8}$$

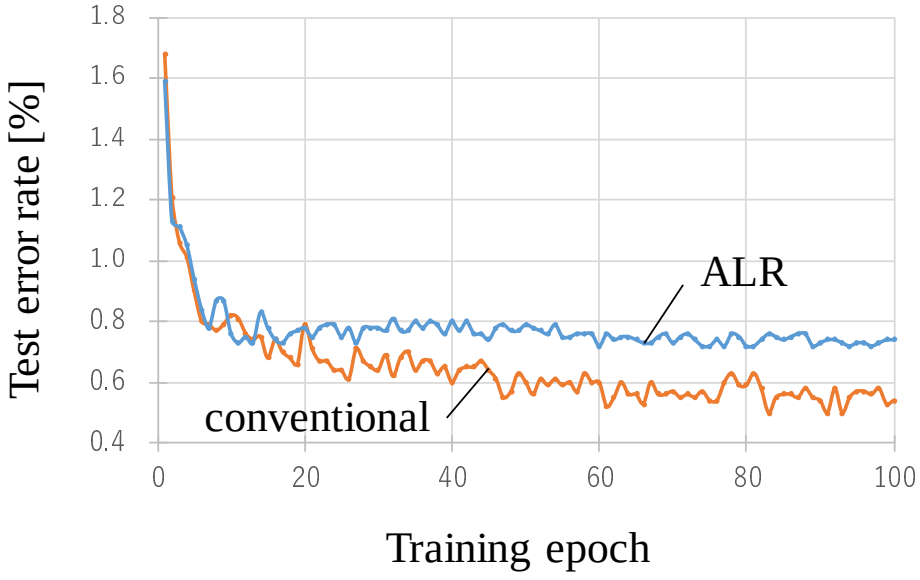
ここで，C はミニバッチのサイズ，K は出力層のユニット数，d_{ck} はサンプル c のユニット k の目標出力，y_k はユニット k の実際の出力，x_c は c の入力，w は重み（およびバイアス）である．ドロップアウト法は，バッチごとにある確率で隠れ層のユニットを一時的に消すことで，テスト精度を改善することができる．ドロップアウト法によって，w がバッチごとに大きく変化するので，$E(w)$ もまた変化する．対数尤度のような他の誤差関数を用いても，同様にドロップアウト法によって影響を受けてしまう．ALR 法はすべての重みに同じ学習率を割り当てるので，そ

のような複雑で非定常な誤差関数においては，大きなスケールファクターは訓練誤差を増加させる傾向にあり，めったに選ばれなくなる．

非定常な誤差関数の効果を軽減するために，ドロップアウト法を用いずに同様



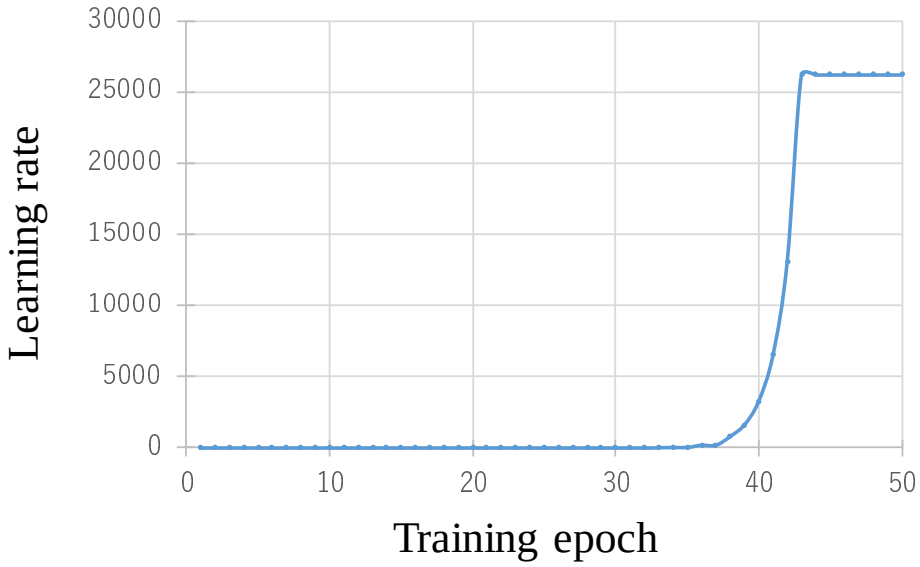
(a) 学習率.



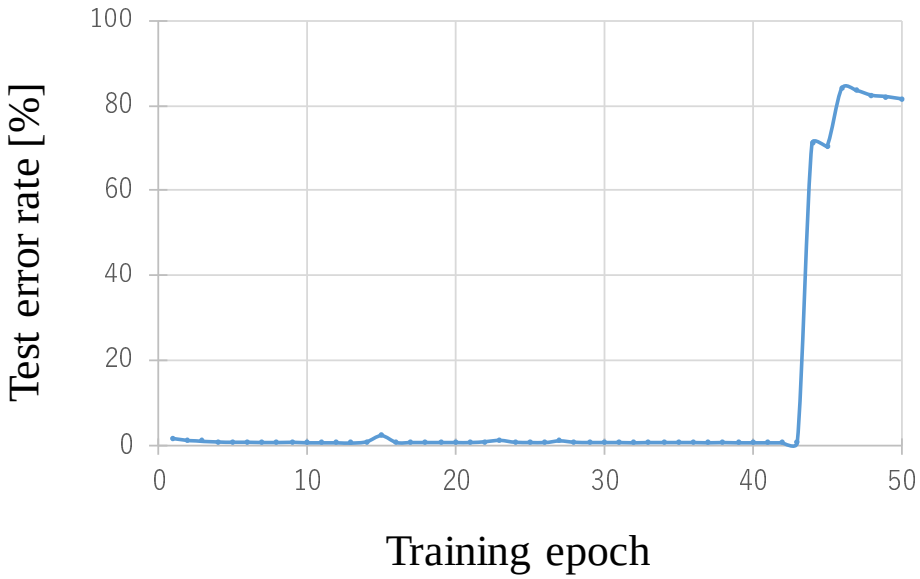
(b) テスト誤識別率.

図 3.7: ドロップアウト法が用いられたときの学習率とテスト誤識別率.

の実験を行った．他の条件はドロップアウト法を用いた実験と同様のものを用いた．図 3.8(a) と図 3.8(b) に示すように，学習率が増加したが，テスト誤識別率は学習中突然大きな値になってしまった．



(a) 学習率.



(b) テスト誤識別率.

図 3.8: ドロップアウト法が用いられなかったときの学習率とテスト誤識別率.

この実験で用いた CNN は MLP よりもずっと多くの重み (1,111,946) を持っており，このことが 2 つのネットワークによる結果の違いを引き起こしたと考えられるかもしれない．そこで，これを調査するために，近い重みの数をもった MLP と CNN を用いて実験を行った．3.4.1 節で用いられた MLP について，隠れ層の数を 1,500 に増やすことで，重みの数を 1,192,510 まで増加させた．この大きな MLP を用いたときの学習率の推移は図 3.5 の推移と同様であった，すなわち，ALR 法が上手く機能した．したがって，CNN が多くの重みを有していることは，CNN における ALR 法の失敗の原因ではなかったといえる．

別の理由として，CNN が，畳み込み層や全結合層などの異なった性質のレイヤーから成っていることが考えられる．それによって，パラメータごとの変化範囲が MLP よりも大きくばらつくかもしれない．もしそうならば，誤差関数の形状は歪んだ谷になる．ALR 法はすべての重みに対し同じ学習率を用いているので，もし学習率がいくつかの重みにとって大きすぎるならば，谷は飛び越えられてしまい，その結果，訓練誤差は増加する．実際，CNN を用いた実験において，学習率が増加したとき，訓練誤差は急激に増加した．

CIFAR-10 データセット（一般物体認識タスクであり，モデルに CNN を要求する）が用いられたとき，学習は同様に失敗した．ALR 法が MLP と CNN 以外のネットワークにも上手く働くかどうかを今後の研究で調査したい．

3.5　パラメータの影響

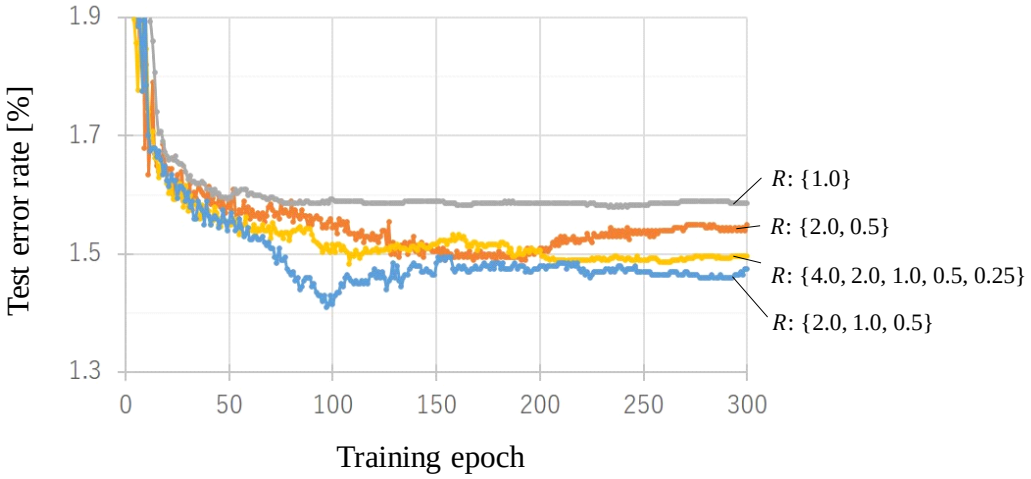
3.5.1　分岐数

分岐数を増加させる（よって，スケールファクターの数も増加する）と，探索することのできるルートの数も増える．しかし，アルゴリズム 2 に示したように，分岐数 N を増加させると，計算コストと重みの記憶量が比例的に増加する．したがって，このパラメータはどちらの要素も考慮して設定されるべきである．

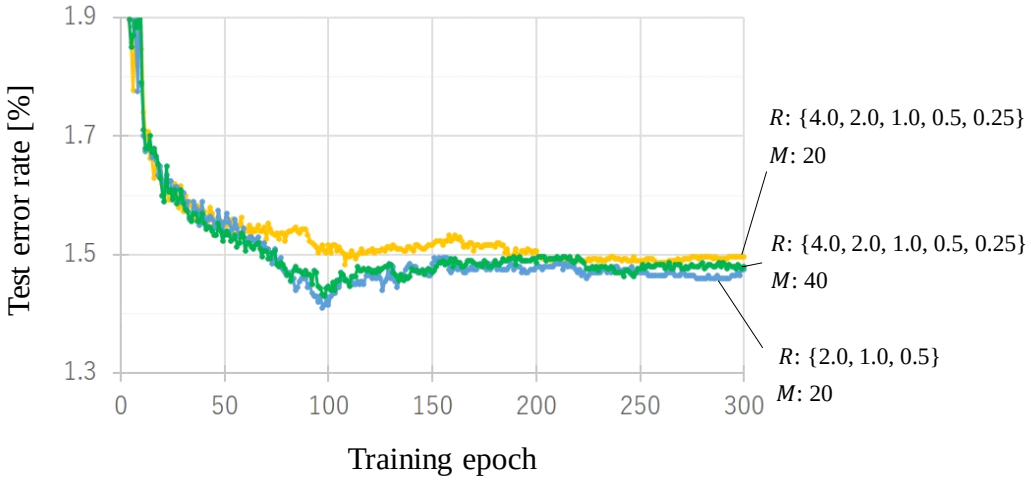
ここで分岐数の影響を実験的に調査した．分岐数は高精度化のためには大きくあるべきだが，計算時間の短縮のためには小さくあるべきである．本実験では分岐数を 1,2,3,5 とし，スケールファクターのセットを a^k の形式で表される 4.0, 2.0, 1.0, 0.5, 0.25 のいくつかの組み合わせとし，ビームサイズを 20 とした．これらのパ

ラメータの値が 1 ならば，ALR 法は従来手法と一致する．試行の回数は 3.4.1 節で行った実験とは異なっている．他の実験条件は 3.4.1 節で述べたものと同じである．

図 3.9(a) に示すように，テスト誤識別率は分岐数が 1 のとき最も大きかった．分岐数が 2 や 3 のときはそれより低かった．しかし，分岐数が 5 のときのテスト誤識別率は，分岐数が 3 のときよりも高かった．これはビームサイズの 20 が，分岐数



(a) ビームサイズ 20 の場合.



(b) ビームサイズが 20 または 40 の場合.

図 3.9: 2 つのビームサイズごとの異なった分岐数を用いたときのテスト誤識別率.R はスケールファクターのセット，M はビームサイズを表す. 分岐数はスケールファクターの要素数に一致する. 異なった初期重みを用いた複数回の試行の平均値が示されている.

が 5 のときに不十分であったからだと考えられる．

分岐数 N とビームサイズ M の関係についてまとめると，以下の 4 つのケースに分けられる．(1)N 大，M 大 : 計算コストは最も大きいが，最も良い性能が得られる．(2) N 大，M 小 : 相対的に大きい訓練誤差をもったノードは保存されないため，悪い局所解を避けることが難しくなり，結果として悪い性能につながる．(3)N 小，M 大 : ALR 法は効率的に用いられるが，上述したようにテスト誤識別率の下限が存在するので，非常に大きな M は計算コストの点から望ましくない．(4)N 小，M 小 : ALR 法は SGD ($N = M = 1$ の場合) に近くなり，したがって ALR 法の利点は小さくなるが，計算コストは最も小さい．

3.5.2　スケールファクター

スケールファクターは学習率が大きく変わるように定義されるべきである．もし学習率が各状態遷移において大きく変化しないならば，ALR 法の効果は非常に小さくなる．すなわち，悪い局所解を避けることは難しくなる．スケールファクターは学習率のみを変化させるので，計算コストには影響しない．

このパラメータの設定において，スケールファクターの集合に，1.0 が含まれるべきであり，それは従来手法を含めることを意味する．他の要素について，1.0 からの違いは大きすぎても小さすぎてもよくない．もし違いが大きすぎるならば，学習時の関数において，各重みは大きく変化する，重みの中には誤差を大きく増加させるものも存在し，そのときすべての重みを含んだ全体の訓練誤差は増加すると考えられる．そのような変化を起こすスケールファクターはめったに選択されない．もし違いが小さすぎるならば，重みは局所解にとらわれがちになり，悪い汎化性能につながる.「大きい」と「小さい」の基準はタスクに依存するので，パラメータチューニングが必要である．

スケールファクターの影響を，a^k の形式によって表される複数のパラメータセットを用いた実験によって調査した．分岐数は 3 に設定し，ビームサイズは 20 に設定した．他の実験条件は 3.4.1 節で述べたものと同じである．

図 3.10 に示すように，テスト誤識別率は，スケールファクターの集合が $\{2.0, 1.0, 0.5\}$ のとき最も低くなった．スケールファクターの要素間により大きな違いがあり，それがより良い解を見つけるためにうまく働いたので，$\{1.25, 1.0, 0.8\}$ の集合を用いたときよりも低いテスト誤識別率が得られた．$\{4.0, 1.0, 0.25\}$ の集合を用い

たときの結果は，4.0 の要素がめったに選択されず，したがってテスト誤識別率の大きな改善につながらなかったので，{2.0, 1.0, 0.5} の集合を用いたときよりも悪い結果となった．

次に，{2.0, 1.0, 0.5} の集合をわずかに変化させた {2.2, 1.0, 0.45} を用いた．2.2 は 2.0 よりも 10%大きく，0.45 は 0.5 よりも 10%小さい．2.2 と 0.45 の積は 0.99 であるので，連続したエポックでそれらの値が両方選ばれると，学習率が 1%減少することになる．それにより学習中の学習率の細かい調整が行うことが可能になり，それは特に学習が収束するときに効果的である．{2.2, 1.0, 0.45} の集合が実験で用いられたとき，最も低いテスト誤識別率は 1.435%であり，それは {2.0, 1.0, 0.5} の集合が用いられたときの結果である 1.410%に比べて大きな違いはなかった．一方，スケールファクターの要素間の違いが大きいとき，前の節で述べたようにテスト誤識別率は大きくなった．したがって，スケールファクターは学習が成功するための，ある適切な幅をもっていると考えられる．

誤差関数はネットワークの構造とデータによって決定され，したがって適切なスケールファクターはそれらに依存する．本研究の実験を通して見つけた一般的なスケールファクターの選択基準は，学習の前半において学習率の増加が十分行われることである．学習率が増加することは，学習の前半において悪い局所解に

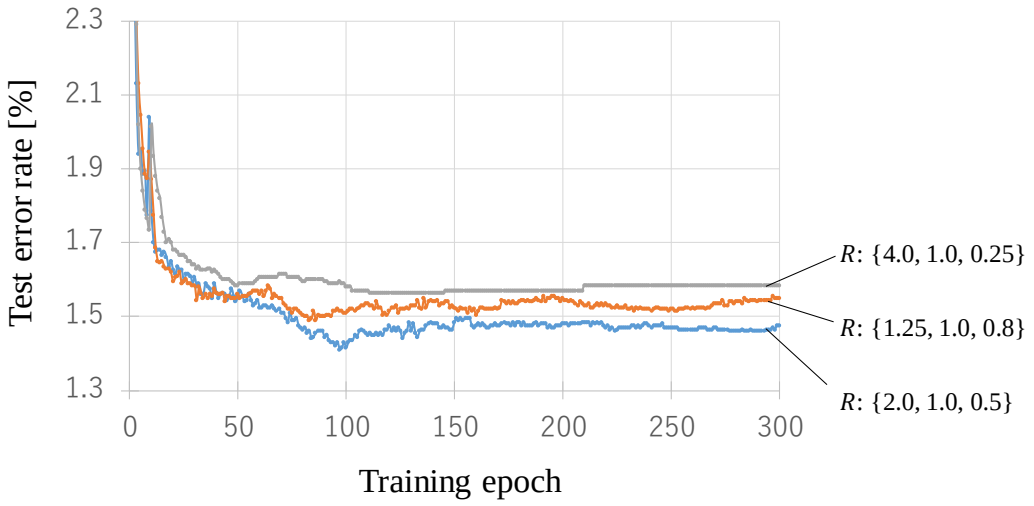


図 3.10: 複数のスケールファクターを用いた実験におけるテスト誤識別率.R はスケールファクターの集合を表す. 異なった初期重みを用いた複数回の試行の平均値が示されている.

とらわれずにより良い解を探索できるために重要である．学習率がほとんど増加しないならば，3.1 節で述べたように ALR 法は従来手法より優れた点がなくなってしまう．

MNIST データセットを用いたとき，その基準の適切さを実証するために，学習の前半における学習率の増加の頻度と学習中の最も小さいテスト誤識別率の間の関係を調査した．ここでは「学習の前半」を「100 エポックまで」と定義した．ここでのエポック数とは，異なったスケールファクターを用いた学習の試行回数ではなく，実際に学習が進んだ回数であるため，スケールファクターとビームサイズを乗じた値である 60 倍にはなっていない．前のエポックからの学習率の変化は，「増加」,「減少」,「変化なし」のいずれかに分類でき,「増加」の回数の割合が算出された．この割合は，学習の前半のエポック数を S 回（本実験では $S = 100$），増加回数を a 回とすると，a/S で表される．複数のスケールファクターの集合を用い，それぞれについて異なった初期重みを用いて学習を 3 回ずつ行った．他の実験条件は 3.4.1 節に述べたものと同じである．

図 3.11 の実験結果は，学習率の増加の回数の割合が 0 または 1 に近いとき，良

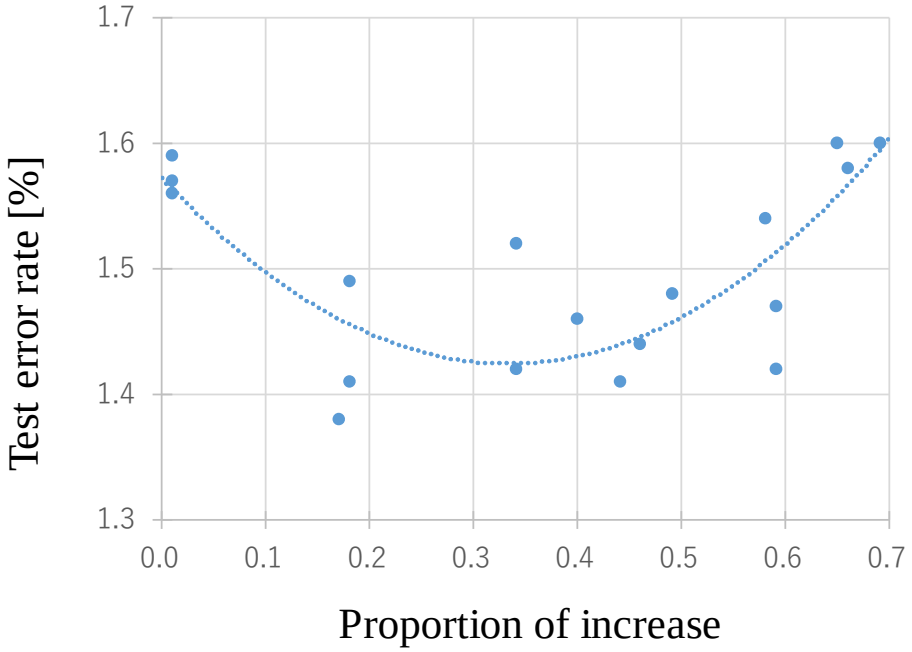


図 3.11: MNIST データセットを用いたときの，学習率の増加の割合とテスト誤識別率の間の関係.

い解が見つけられなかったことを示しており，そのときのスケールファクターは適切ではなかったと考えられる．増加の回数の割合が 0 に近いとき（例えば {4.0, 1.0, 0.25} のスケールファクターによって得られる），パラメータは悪い局所解にとらえられたと考えられる．また，増加の回数の割合が 1 に近いとき（例えば {1.001, 1.0, 0.999} のスケールファクターによって得られる），学習率が非常にゆっくり増加したので，パラメータは十分に広い領域を探索することができなかったと考えられる．増加率が 0.1 と 0.6 の間のとき，小さいテスト誤識別率が得られた．3.4.1 節で用いた {2.0, 1.0, 0.5} の集合は，エポック 100 までの増加率の平均が 0.18 だったので，この基準を満たしていた．

3.5.3　ビームサイズ

もしビームサイズ M が 1 に設定されるならば，最も低い訓練誤差よりも高い誤差をもつノードは除去されてしまう．しかし，たとえノードがあるエポックでより高い誤差をもっているとしても，それはその後のエポックにおいて最も低い訓練誤差をもつかもしれない．したがって，悪い局所解を避けるためにこのパラメータを大きな値に設定することは重要である．理想的には，このパラメータはすべての状態が保存できるようにするため，制限されないのが良い．しかし，アルゴリズム 2 に示したように，ビームサイズの増加は計算コストと重み記憶の数の比例的な増加につながる，したがって，このパラメータは両方の影響を考慮して定義されるべきである．

時間が十分経過したとき，木の先端には M 個のノードがあり，それぞれ N 個の子ノードを生成し，それぞれについて学習の処理を行うのでその時間計算量は計算量は $O(MN)$ である．総エポック数を T とすると，この計算が T 回繰り返されるので，これはは $O(MNT)$ となる．

私たちはビームサイズの影響を実験的に調査した．分岐数と同様に，ビームサイズもまた精度の向上のためには大きいほうが良いが，計算時間の短縮のためには小さいほうが良い．分岐数は 3 に，スケールファクターの集合は {2.0, 1.0, 0.5} に設定した．他の実験条件は 3.4.1 節で述べたものと同じである．

図 3.12 に示すように，ビームサイズが 1 のとき，重みが悪い局所解にとらわれる傾向があり，テスト誤識別率は最も高かった．ビームサイズが 20 または 40 のとき，最も低いテスト誤識別率が得られた．これらの 2 つのビームサイズを用いた

とき，ビームサイズ 20 がこれらの条件下で最も低い誤識別率を得るために十分であるので，同じ結果になった．したがって，ビームサイズを増やすことによって，より広い領域が探索されるといえる．これにより，悪い局所解を避けることが可能になるが，この効果には上限が存在する．

図 3.13 に示すように，学習率の推移の傾向はビームサイズによって影響を受ける．ビームサイズが 1 または 2 のように小さいとき，状態の記憶数が不十分であるので，重みは悪い局所解にとらわれる傾向にある．ビームサイズが 4 または 5 のように大きいとき，学習率の推移はビームサイズが 20 のとき（図 3.5）と同様の傾向であった．すなわち，高い学習率が学習中得られた．

3.6　他のデータセットを用いた実験

ALR 法が広く応用可能であることを確かめるために，他のデータセットを用いた実験を行った．UCI Machine Learning Repository(Lichman , 2013) に含まれる多クラス分類用のデータセットである，Car Evaluation データセット, Wine データセット, および Letter Recognition データセットを MLP とともに用いた．

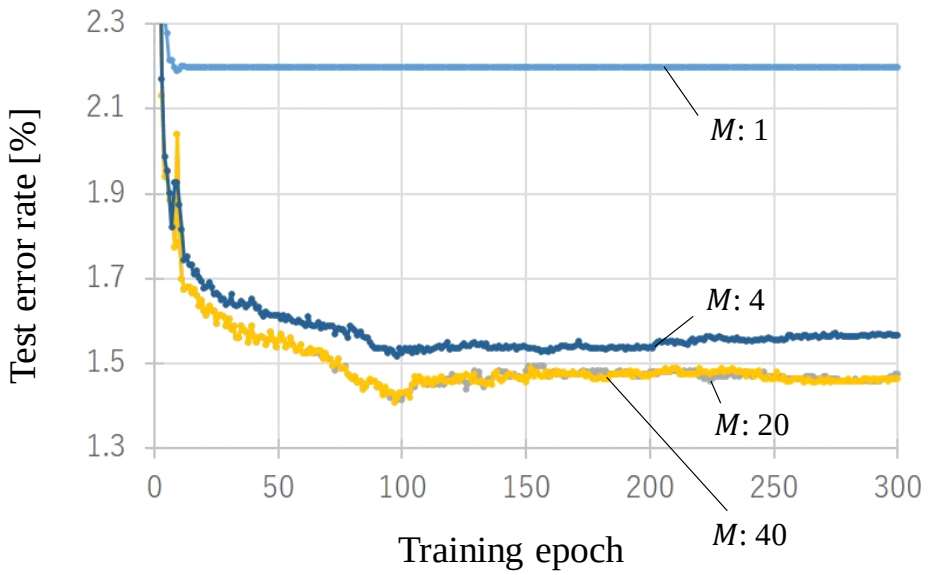


図 3.12: 異なったビームサイズ (ビームサイズ: 1, 4, 20, 40) を用いたときのテスト誤識別率.M はビームサイズを表す.

表 3.5: 他のデータセットを用いた実験結果.「ALR」と「固定」の列にある括弧外の数字，および「Adam」の列にある数字は，異なった初期学習率を用いた 10 回の試行における最も低いテスト誤差を示している.「ALR」と「固定」の列にある括弧内の数字は，そのテスト誤差が得られたときの初期学習率を示している.

データセット	ネットワーク構造	ALR	固定	Adam
Car	6-10-4	0.076(lr:0.7)	0.079(lr:0.6)	0.103
Wine	13-10-3	0.062(lr:0.8)	0.094(lr:0.7)	0.116
Letter	16-100-26	0.209(lr:0.3)	0.220(lr:0.4)	0.258

Car Evaluation データセットは 1,728 サンプルの自動車に関するデータを含んでおり，サンプルは全体の評価である 4 つのクラスに分類される．各サンプルは，値段やドアの数といった 6 個の特徴をもっている．Wine データセットは 178 サンプルを含んでおり，各サンプルは 3 種類のワインのどれかに分類される．特徴の数は，アルコールやリンゴ酸など，全部で 13 である．これらのデータセットにおいて，私たちはテストデータとしてサンプルの 35%を利用した．Letter Recognition データセットは 20,000 サンプルを含んでおり，公式で推奨されているように，最初の 16,000 サンプルを訓練データとして，残りの 4,000 サンプルをテストデータとして用いた．サンプルは英語のアルファベットである A から Z までの 26 個のクラスに分類される．特徴の数は，統計的モーメントやエッジの数など，全部で 16 である．

分岐数は 3 に，スケールファクターのセットは $\{1.001, 1.000, 0.999\}$ に，ビームサイズは 20 に設定した．このスケールファクターのセットは，後で調査結果を示す一般的なスケールファクターの選択基準に従っている．スケールファクターの各要素の値が近いため，結果に与える初期学習率の影響がより大きくなる．ALR 法と従来手法の両方において，10 回の試行が異なった初期学習率（0.1 から 1.0 まで 0.1 おきに）を用いて行った．これらの結果の中で最も良い結果を比較した．小さなデータセット，すなわちテストサンプルの数が少ない場合，テスト誤識別率は粗い評価になるため，3.4.5 節の式 (3.7) に示した誤差関数をもつテスト誤差を評価のために用いた．さらに，3.4.3 節の実験で用いたものと同じ一般的なパラメータを用いた Adam も，比較のために利用した．

実験結果を表 3.5 に示す.「ALR」と「固定」の列にある括弧外の数字，および

「Adam」の列にある数字は，異なった初期学習率を用いた 10 回の試行における最も低いテスト誤差を示している.「ALR」と「固定」の列にある括弧内の数字は，そのテスト誤差が得られたときの初期学習率を示している．さらに，これらの実験で用いられたニューラルネットワークの各層のユニット数を「ネットワーク構造」の列に示している．すべてのデータセットを用いた実験において，ALR 法が最も低いテスト誤差を示した．

3.5.2 節の最後に行った実験と同様に，学習の前半における学習率の増加の頻度と，学習中の最も小さいテスト誤差の関係を調査した．この節で用いられたデータセットを代表して，Car Evaluation データセットを用いた．図 3.14 に示した実験結果は，学習率の増加の割合が 0 に近いとき，良い解が見つけられなかったことを示しており，そのときのスケールファクターが適切ではなかったといえる．MNIST データセットを用いた実験の結果と同じように，学習の前半における学習率の増加が小さなテスト誤差を得るために重要である．この節の実験で用いた {1.001, 1.0, 0.999} のスケールファクターは，100 エポックまでの増加の割合の平均が 0.13 だったので，3.5.2 節で述べたスケールファクター選択基準を満たしている．

{2.0, 1.0, 0.5} のスケールファクターが MNIST データセットを用いた実験では上手く働いたが，このデータセットでは，エポック 100 までの増加の割合の平均が 0.03 だったので，基準を満たさなかった．そのスケールファクターにおける 0.5 の要素が最初から最後まで集中的に選択され，したがってパラメータが悪い局所解にとらわれたと予測される．

3.7　結論

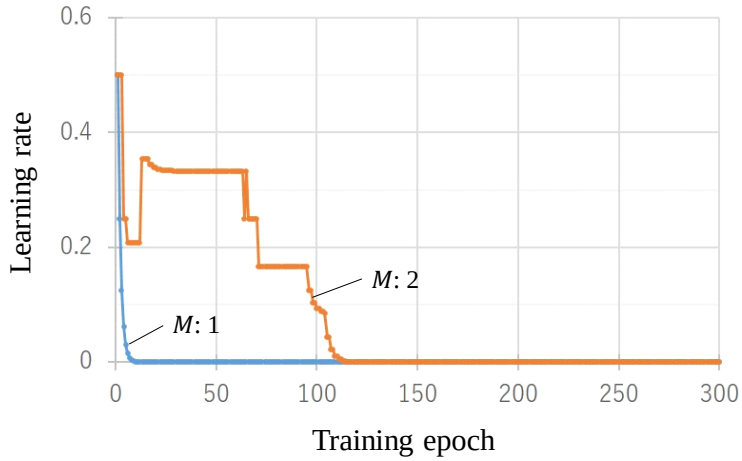
ビームサイズを用いた木探索にもとづいて，ALR 法はできるだけ訓練誤差を減らすように学習率を適応的に増加または減少させる機械学習アルゴリズムとして設計され，それは学習率を主に減少させる手法とは異なり，悪い局所解を避け，良い解を効果的に見つけることを可能にした．

実験結果は，提案手法が異なったデータセットでテスト誤識別率を下げることができることを示した．特に興味深い点は学習率が大きく増加し，結果として，悪い局所解につながる領域の外で良い解が見つけられるということである．さらに，提案手法は初期学習率にあまり依存していないという点でより頑強である．

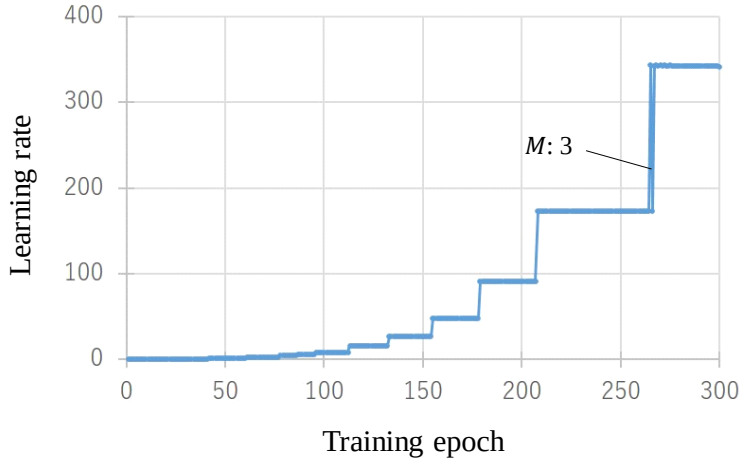
つぎに，3 つの主要パラメータ（分岐数，スケールファクター，ビームサイズ）の影響を調査し，分岐数とビームサイズは上限が存在し，その範囲で，計算時間とメモリ容量を考慮して，十分大きくとるべきであることを示した．スケールファクターについては，一つの基準として学習率の増加の回数の割合を 0.1〜0.6 程度とすることを提示した．

ALR 法を用いて畳み込みニューラルネットワークを訓練したが，必ずしも望ましい結果は得られなかった．その理由を明らかにし，手法を改良することは今後の課題である．

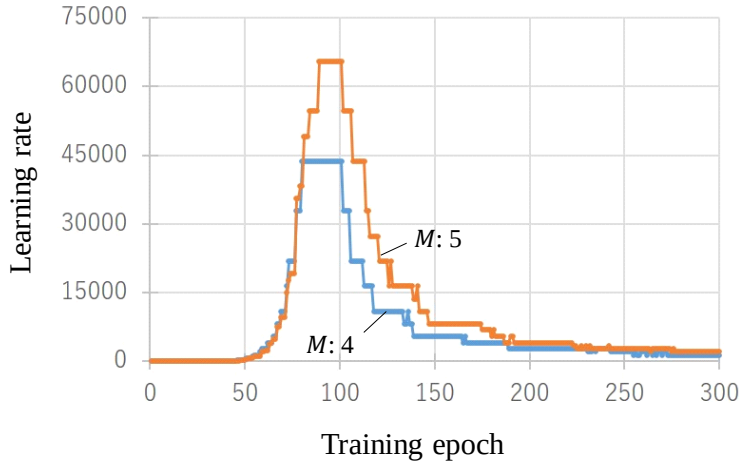
ALR 法において，すべての重みは同じ学習率を用いて更新されている．AdaGrad のように，個々に学習率を制御するための効率的な手法へと拡張することもまた今後の課題である．



(a) 学習率 (ビームサイズ: 1, 2).



(b) 学習率 (ビームサイズ: 3).



(c) 学習率 (ビームサイズ: 4, 5).

図 3.13: 異なったビームサイズ (ビームサイズ: 1, 2, 3, 4, 5) を用いたときの学習率. 学習率の推移の傾向はビームサイズに依存することがわかる.M はビームサイズを表す.

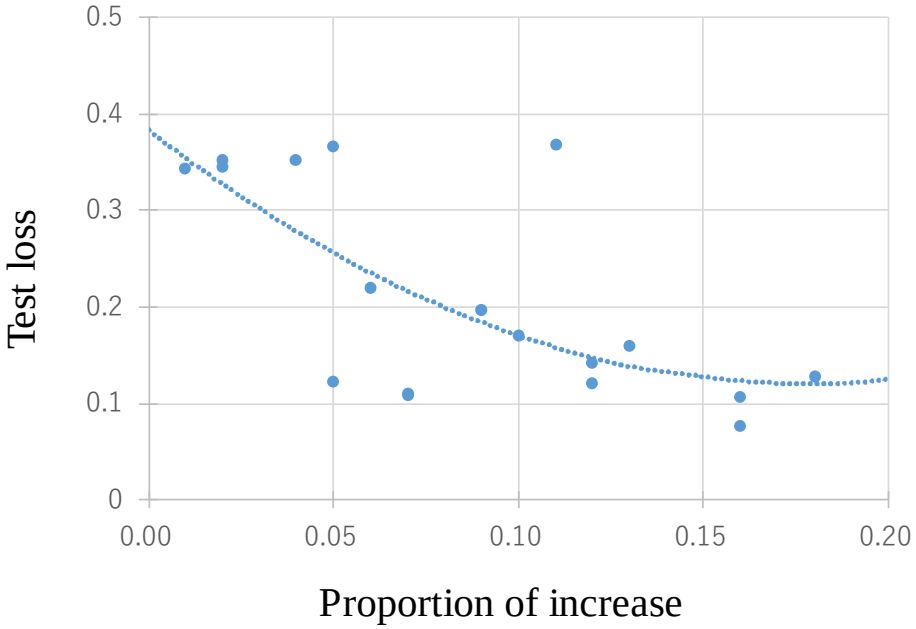


図 3.14: Car データセットを用いたときの，学習率の増加の割合とテスト誤差の関係.

第4章　確率的最適化とバッチサイズ増加法

4.1　導入

より良い解は一般的により新しいデータの正確な予測へとつながるので，非凸最適化法は，機械学習の重要な要素の一つである．目的関数は，一般に多数の凹凸構造をもった複雑な形状をしているので，最小化することは難しい．

解析的な方法ではほとんど求められないので，数値的最適化手法が欠かせない．非凸問題における代表的な数値最適化手法は，式 (2.11) にも示した勾配降下法である：

$$\theta_{k+1} = \theta_k - \eta \cdot \nabla E(\theta_k), \tag{4.1}$$

ここで，θ_k はステップ k でのパラメータで，$E(\theta)$ は損失関数，η は更新量を調整するための値である学習率を示している．誤差は，各点における勾配の逆方向にパラメータ θ を動かすことによって減らされる．大域解への収束は保証されておらず，変数はしばしば悪い局所解や鞍点にとらわれてしまい，その結果，汎化性能が悪化してしまう．

望ましくない解を避けるための複数の手法がこれまでに提案されてきた．例えば，AdaGrad (Duchi, Hazen, & Singer, 2011)，RMSProp (Tieleman & Hinton, 2012)，AdaDelta (Zeiler, 2012)， Adam (Kingma & Ba, 2015) などである．これらの手法は勾配降下法にもとづいており，ニューラルネットワーク学習においてよく用いられる．これらは学習中の更新量を調整し，ステップサイズを減らすので，序盤は解の広い範囲の探索が行われるが，パラメータはより安定して収束することが可能になる．

悪い局所解を避けるための一つの単純な方法は，異なった初期値から探索を始め，最も低い誤差をもった解を選ぶということである．異なった初期値を用いると異なった解が得られることがあるので，初期値は重要である．このアプローチは

上手く働くが，繰り返し試行を行う必要があり，時間がかかってしまう．これまで複数の研究者が最適化問題の初期化の重要性に焦点を当ててきた (Sutskever et al., 2013; Mishkin & Matas, 2015).

別のアプローチは，アルゴリズムがランダム性を用いて非決定的に解を選ぶようなランダム化を用いることである．特に，シミュレーテッド・アニーリング (Kirkpatrick, Gelatt, & Vecchi, 1983; Černý, 1985; Rere, Fanany, & Arymurthy, 2015) は大域的最適化問題のための強力なアルゴリズムである．それは，時間とともに減少する温度パラメータによって定義された確率によって，関数値が増加する方向にもパラメータを動かす．このアルゴリズムの詳細は 2.6 節で述べた．

シミュレーテッド・アニーリングは勾配の計算できない関数においても利用可能な汎用的な手法であるため，勾配の計算できるニューラルネットワークでは実用的に使用されない．しかし，シミュレーテッド・アニーリングのアイデアを一般化することは非凸最適化問題のための他の効果的な手法の開発につながり得る．したがって，本論文では，シミュレーテッド・アニーリングに基づいた包括的な枠組みを与える．それは，ランダム性が戦略的に用いられるというものである．具体的には，ランダム性は探索の前半においては強く，探索の間徐々に弱くする，すなわち，探索の戦略が非定常から定常へと変化する．本論文では，この枠組みに基づいた方式を，ニューラルネットワークのミニバッチ学習へと適用する．

本論文の残りは以下のように構成される．4.2 節では準備および動機について述べる．4.3 節では，非凸最適化問題のための効果的な探索アプローチの包括的な枠組みを述べる．4.4 節では，なぜニューラルネットワークにおいて大きなバッチサイズを用いた学習が汎化性能を悪化させるか (Keskar et al., 2017) を説明し，上記の枠組みをミニバッチ学習に適用する．4.5 節で行った実験の結果は，その枠組みの妥当性を支えるものとなる．

4.2　準備と動機

非凸最適化問題は，損失関数が高次元空間においてしばしば複雑な形状をもつので，機械学習における重要な課題である．山登り法やシミュレーテッド・アニーリング，タブーサーチ (Glover, 1986)，蟻コロニー最適化 (Bonabeau, Dorigo, & Theraulaz, 1999)，遺伝的アルゴリズムといった多くの探索アルゴリズムが，複雑な損失関数

をもった問題におけるより良い解を見つけるために提案されてきた．

損失関数の形状について良く知っておくことが，悪い局所解を避けるために要求される．多くの理論的もしくは実験的な研究のおかげで，損失関数の形状についての複数の事実が明らかにされてきた．

理論的な成果の例として，Kawaguchi (Kawaguchi, 2016) はすべての局所解は大域解であり，大域解でないすべての停留点は鞍点であるということを，いくつかの仮定のもとで証明した．しかし彼は，その仮定が強かったので，悪い局所解はディープな非線形ニューラルネットワークでは実際に生じるだろうと結論付けた．Lu と Kawaguchi (Lu & Kawaguchi, 2017) は，ニューラルネットワークの非線形性を考えなければ，ネットワークの深さは損失関数の非凸な形状を作るけれども悪い局所解を生成しないということを証明した．しかし，彼らは活性化関数の非線形性によって引き起こされる非凸形状の議論を行わなかった．したがって，悪い局所解を避けるための探索手法は，依然として要求される．

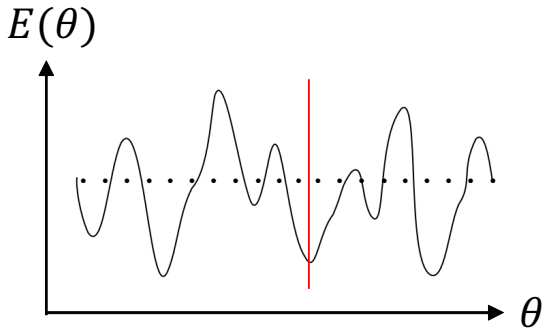
特に注目すべき実験的な成果の例として，Keskar ら (Keskar et al., 2017) によって仮定された図 2.4 に示したシャープな局所解の存在についての議論がある．損失関数は一般に複雑であるので，シャープな局所解の存在を証明することは難しく，Keskar らは実験的にそれを実証しようとした．彼らは大きなミニバッチが学習で用いられるときに，パラメータがシャープな局所解にとらわれると主張した．

しかし，Dinh らはそのようなシャープな局所解の概念が上手く定義されていないと考えた (Dinh et al., 2017)．彼らは，もし Keskar らのシャープさの定義に従うならば，損失関数は自由に変形でき，それは図 4.1 に示すように，すべての局所解が再パラメータ化によってフラットにすることができるということを意味するので，シャープさおよびフラットさに対する Keskar らの定義が適切ではなかったと主張した．彼らはシャープさとフラットさの定義を再考する必要があると結論付けた．

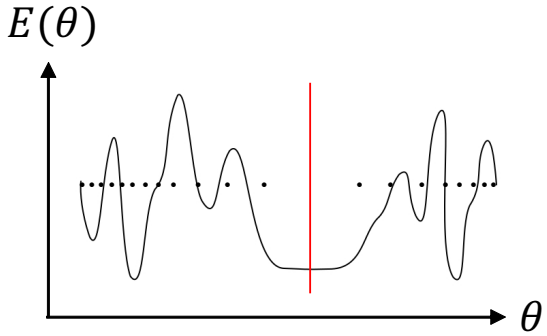
4.3　包括的な枠組み

以前の節で述べたように，ニューラルネットワークにおいて大きなバッチサイズを用いた学習は汎化性能の悪化という結果になる傾向にある (Keskar et al., 2017)．この問題を避けるために，この節では，非凸最適化問題を解くための効果的なア

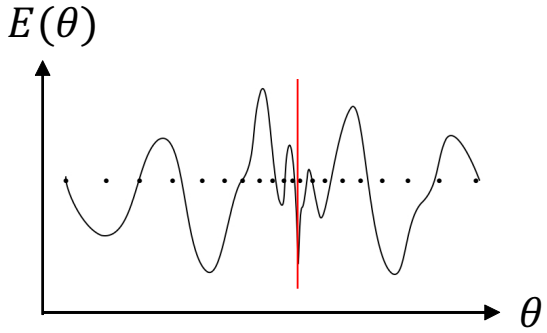
プローチの包括的な枠組みを与える.



(a) 元々のパラメータ化



(b) 再パラメータ化



(c) 別のパラメータ化

図 4.1: 損失関数の形状が選ばれたパラメータ空間にどれくらい依存するかに関する一次元の例 (Dinh ら (Dinh et al., 2017) に基づく). θ 軸はパラメータの値を表し, E 軸は誤差を表す.

4.3.1　ノイズを用いた更新

更新が下方向（すなわち関数値が減少する方向）に進む標準的な勾配降下法（式 (4.1)）を用いて悪い局所解を避けることは難しい．よって，悪い局所解を避けるための効果的な方法は，時々ランダムな方向に更新が進むように，式 (4.1) における $\nabla E(\theta_k)$ を調整することである．したがって，探索ステップ t に依存したノイズ ξ_t がよく $\nabla E(\theta_k)$ に加えられる．それは，加えられたノイズにより作られるランダム性のため，更新が必ずしも下方向に進むとは限らないことを意味する．このランダムな移動は悪い局所解を避けるために大きく貢献する．

しかし，そのような方向への更新は収束を遅くするか，あるいは難しくするので，そういったノイズを単純に付け加えることは，必ずしも上手く働かない．したがって，悪い局所解を避けることと安定した収束の間にはトレードオフの関係がある．

したがって，付け加えたノイズのランダム性の程度は，以下のように探索プロセスの段階に応じて制御されるのがよい．探索の前半では，より良い解を探索することが，収束を行うことよりも重要である．よって，大きな ξ_t が $\nabla E(\theta_k)$ に加えられるべきである．それから探索が進むにつれて，目標の解への収束がより重要になってくるので，ξ_t はランダム性の影響を弱めるために減らされるのがよい．ξ_t が 0 に近づくとき，更新手法は標準的な勾配降下法とほぼ同じである．

4.3.2　ノイズ減少戦略

ノイズの作り方は各探索手法によって異なるが，ノイズは探索ステップ t でのキーパラメータ ϕ_t によって制御されるべきであり，すなわち，ノイズは $\xi(\phi_t)$ によって与えられる．シミュレーテッド・アニーリングにおいて，ϕ_t は以下に示す状態確率 p に一致する．

$$p = \exp\left(-\frac{E(\theta') - E(\theta_t)}{\mathrm{Temp}_t}\right) \tag{4.2}$$

ここで，θ' は，$E(\theta') > E(\theta_t)$ となるような θ_t の次の探索点の候補の状態であり，確率 p で θ' が次の状態として実際に採用される．

探索中の各変数の変化を図 4.2 に示す．探索が進む（t が無限大に近づく）につれて，ϕ_t は定数 (α) に，ノイズ $\xi(\phi_t)$ は 0 に近づき，そして探索戦略は式 (4.1) の標

準的な勾配降下法に近づく．シミュレーテッド・アニーリングにおいて，α は 0 に一致する．それは更新が決して上方向に進まないことを意味する．温度 Temp_{t+1} は徐々に減少するので，p は 0 に近づく．それはシミュレーテッド・アニーリングがこの枠組みに従うことを意味する．

4.4　ニューラルネットワーク学習への適用

この枠組みの応用として，ニューラルネットワークにおける訓練誤差の最小化問題に焦点を当てる．ニューラルネットワーク，特にディープニューラルネットワークの損失関数は一般に非常に複雑であり，したがって最小化するのが難しい．

Keskar ら (Keskar et al., 2017) は，大きなミニバッチを用いた学習は，シャープな局所解をもつ領域にとらわれる傾向があるので，汎化性能を悪化させ得ると主張した．Dinh ら (Dinh et al., 2017) は（うまく定義されていない）フラットさの概念において，重大な問題があると述べたが，大きなバッチサイズが汎化性能を悪化させるという Keskar らの実験による発見は，注目に値する．悪化の原因はシャープな局所解にとらわれたことであるかもしれないが，なぜ大きなバッチサイズがそのような結果につながるのかの理由は Keskar らによって議論されなかった．本論文では，この現象が損失関数の定常性によって説明できると考え，4.3 節で述べた枠組みに従い，悪い局所解を避けるための新しいアプローチを与える．

Update number:	t	$\longrightarrow$	∞
Key parameter:	ϕ_t	$\longrightarrow$	α
Noise:	$\xi(\phi_t)$	$\longrightarrow$	0
Update direction:	$\nabla_\theta(E(\theta_k)+\xi(\phi_t))$	$\longrightarrow$	$\nabla E(\theta_k)$

図 4.2: 探索中の各変数の変化.

4.4.1　ミニバッチ確率的勾配降下法

ミニバッチ確率的勾配降下法 (MSGD) はニューラルネットワーク学習において広く用いられている．そのアルゴリズムをアルゴリズム 3 に示す．サンプル番号の範囲は 0 から $M-1$ までであり，ミニバッチ内のサンプル番号の範囲である $(i;M)$ と $(i;i+N)$ は，それぞれ $[i,M)$ と $[i,i+N)$ を意味する．ここで，各ミニバッチ内のサンプルは，すべての訓練サンプルが 1 つのエポック内で重複なしで用いられるように選ばれる．

Algorithm 3 ミニバッチ確率的勾配降下法

Require: 全更新回数: S
　　バッチサイズ: N
　　訓練データ数: M
$i \leftarrow 0$
for t from 0 to $S-1$ **do**
　if $i+N \geq M$ **then**
　　$\theta \leftarrow \theta - \eta \cdot \nabla_\theta E(\theta; x^{(i;M)}; y^{(i;M)})$
　　$i \leftarrow 0$
　else
　　$\theta \leftarrow \theta - \eta \cdot \nabla_\theta E(\theta; x^{(i;i+N)}; y^{(i;i+N)})$
　　$i \leftarrow i+N$
　end if
end for

ニューラルネットワークの安定性を論じる前に，MSGD の損失関数のための便利な置き換えを示す．MSGD における損失関数の勾配，すなわち，更新量は，各サンプルの更新量を用いて以下のように与えられる．

$$\frac{\partial E_t}{\partial \theta} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E^{(n)}}{\partial \theta}, \tag{4.3}$$

ここで，θ はパラメータのセットを，E_t はバッチサイズが N である t 番目のミニバッチによる損失関数を，$E^{(n)}$ はミニバッチ内の n 番目のサンプルの損失関数を表す．したがって，$\partial E_t/\partial\theta$ は t 番目のミニバッチの更新量の平均であり，式 (4.1) における $\nabla E(\theta_k)$ として，勾配降下法に利用される．MSGD において，ミニバッチごとに 1 回のみの更新が，$\partial E_t/\partial\theta$ を計算することで行われる．

式 (4.3) の右辺は以下の式によって変形できる．

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial E^{(n)}}{\partial\theta}=\frac{\partial}{\partial\theta}\left(\frac{1}{N}\sum_{n=1}^{N}E^{(n)}(\theta)\right). \tag{4.4}$$

これは各損失関数の勾配の平均が，損失関数の平均の勾配と同じであることを意味する．したがって，損失関数の平均は，以下の式によって与えられるように，更新量の平均の代わりとして，ミニバッチ学習における更新のために用いることができる．

$$E_t(\theta)=\frac{1}{N}\sum_{n=1}^{N}E^{(n)}(\theta). \tag{4.5}$$

これは，ミニバッチ学習における損失関数の振る舞いが議論されるときに便利である．

4.4.2　損失関数の定常性

本研究では，大きなバッチサイズが損失関数を定常にする，すなわち関数の形状が更新ごとにほぼ同じであると推測する．もし損失関数が定常ならば，悪い局所解から逃れることは難しい．なぜなら，そのような解の周りの形状がそれほど変化しないからである．もし損失関数が非定常ならば，その形状が更新ごとに変化するので，悪い局所解から逃れることはより簡単である．

大きなバッチサイズを用いるとき，なぜ損失関数が定常になるかを考える．ここで，損失関数の形状は多数のレイヤーおよび非線形な活性化関数のために非常に複雑なので，解析することができないから，損失関数の形状の定常性の代わりに損失関数の値の定常性を評価する．非定常な損失関数は誤差の値を大きく変えるので，この値は損失関数の定常性を調べるために利用することができる．

4.4.1 節で議論したように，損失関数の勾配の平均は損失関数の平均の勾配と同じである．すなわち，バッチサイズ N を用いた時の誤差 $E_N(\theta)$ は，$(1/N)\sum_{n=1}^{N}E^{(n)}(\theta)$ によって与えられる．したがって，小さなバッチサイズ N_s を用いたときの誤差 $E_s(\theta)$ および大きなバッチサイズ N_l を用いたときの誤差 $E_l(\theta)$ は以下の式によって与え

られる．

$$E_s(\theta) = \frac{1}{N_s}\sum_{n=1}^{N_s} E^{(n)}(\theta) \tag{4.6}$$

$$E_l(\theta) = \frac{1}{N_l}\sum_{n=1}^{N_l} E^{(n)}(\theta). \tag{4.7}$$

各ミニバッチの誤差 E の分散を $\mathrm{var}(E)$ と表すことによって，$\mathrm{var}(E_s(\theta)) > \mathrm{var}(E_l(\theta))$ を証明することができる．

すべてのサンプルが一つのバッチを構成する特別な場合を考え，すべての M サンプルの誤差の平均を $(1/M)\sum_{n=1}^{M} E^{(n)}(\theta)$ と表す．ここで，これを「真の誤差」と呼ぶことにする（真の誤差は本来，訓練データがサンプリングされる母集団の分布の誤差の平均のことを指すが）．各バッチは訓練データから独立にサンプリングされる．各バッチの誤差の分散を表した概念図である図 4.3(a) に太線で示すように，もしすべての訓練データを用いたときの真の誤差 μ が一定ならば，各バッチの誤差の平均は，平均 μ・分散 σ^2/N の分布に従う．$N_s < N_l$ であるので，$\sigma^2/N_s > \sigma^2/N_l$ であり，したがって，$\mathrm{var}(E_s(\theta)) > \mathrm{var}(E_l(\theta))$ が成り立つ．

ニューラルネットワークにおいては，図 4.3(b) に太線ですべての訓練データを用いた真の誤差を表しているように，学習中，誤差が勾配降下法によって減少するので，この関係はそれほど単純ではない．真の誤差は，t が増加するにつれて，減少する傾向にある．この場合，μ_t が変化するので，E_t は平均 μ の分布に従わない．

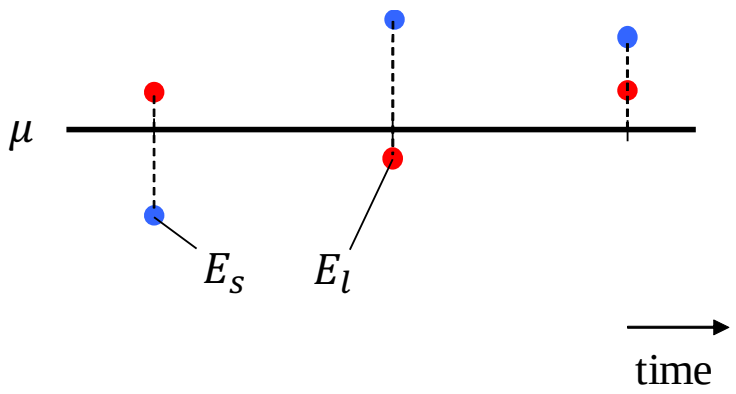
しかし，学習中の誤差の減少は，それがニューラルネットワーク学習の目的であるので，必要なことである．したがって，μ と各バッチの平均誤差との差 (図 4.3(a)) ではなく，μ_t と各バッチの平均誤差との差 (図 4.3(b)) の分散が重要である．図 4.3(a) の誤差が一定の場合と同様に，差の分散は平均 μ_t の分布に従う．したがって，$\mathrm{var}(E_s(\theta) - \mu_t) > \mathrm{var}(E_l(\theta) - \mu_t)$ という関係が得られる．この不等式が成り立つとき，本論文では，右辺の損失関数は，左辺の損失関数よりも，“より定常” であると捉える．

一般に，固定された損失関数上で，データが未学習の状態であるとき，局所解から逃れ，より小さな誤差を探し求めることは，悪い局所解にとらわれているよりも，より良い汎化性能を生み出すことができる．一方，過学習の状態においては，そのような脱出および探求は，より悪い汎化性能を生み出し得る．よって，過学習中に局所解にとらわれることは，さらなる過学習を避けるので，より良い汎

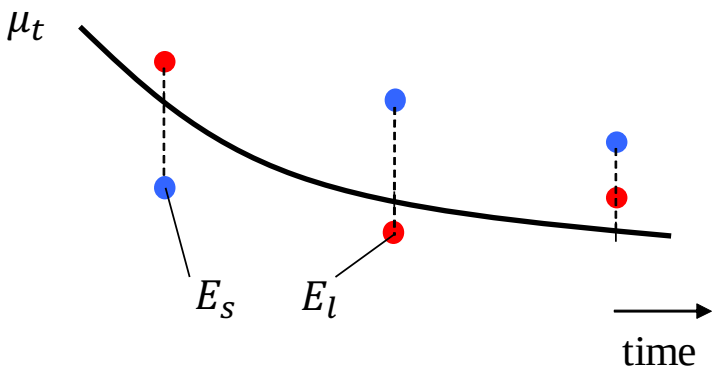
化性能を生み出すかもしれない．したがって，適した戦略は，未学習中は広い範囲を探索することによってより良い局所解を見つけ，それから過学習中は局所解に安定して収束しようとすることであると考えることができる．

4.4.3　可変バッチサイズ

前節での安定性についての議論に基づき，小さいミニバッチの使用はより良い解の発見にとってより適切であり，大きいミニバッチの使用は安定した収束をするためにより適切であると考える．ミニバッチ学習に対する私たちのアプローチは，この予測に基づいた戦略を効果的に用いる．より正確に述べると，以下のようにバッチ学習に対して，4.3 節で議論した枠組みを適用する．ここで，図 4.2 の



(a) 真の誤差が固定される場合.



(b) 真の誤差が減少する場合.(実際のニューラルネットワーク学習)

図 4.3: 各バッチにおける誤差の分散に関する模式図.

キーパラメータ ϕ_t によるノイズ制御と，t 回目の更新におけるバッチサイズを結びつける．すべての訓練データがバッチごとに使用される場合，誤差関数は一様であり，その勾配を「真の勾配」と考える．サンプルが訓練データから選ばれるとき，バッチごとに計算された勾配は異なっており，そのような真の勾配との差のそれぞれを，真の勾配に加えられたノイズ $\xi(\phi_t)$ と考える．勾配の変化は，結果として誤差の変化となるので，ノイズ $\xi(\phi_t) = |E_t(\theta) - E_M(\theta)|$，すなわち訓練誤差と真の誤差の違いを設定することで，私たちの枠組みにこの考えを組み込む．

ノイズの減少は，バッチサイズの増加，したがって損失関数を非定常から定常へと変化させることによって達成される．これは，探索中の探索の目的を，より良い解を見つけることから安定した収束を得ることへと変化させることになる．この戦略は非常に単純で，実装するのが容易である．それはバッチサイズを，前もって決めた初期サイズと最終サイズを用いて，学習中線形に増加させるだけである．バッチサイズを徐々に増加させることによって，バッチサイズがまだ小さいときに，未学習の状態において大きなノイズを用いてシャープな局所解から逃れることを期待することができる．バッチサイズが大きくなってきたら，探索は徐々に安定的に，過学習の状態においてさえも小さなノイズを用いてフラットな局所解へと収束する傾向にある．したがって，最終的に，誤差は小さいが汎化性能は悪いであろうシャープな局所解にとらわれるよりも，良い汎化性能が得られることが期待できる．

この手法は，真の勾配からの違いに焦点を当てているという点において，Stochastic Average Gradient(SAG) (Roux, Schmidt, & Bach, 2012) に似ている．SAG は，SGD のように各イテレーションでランダムに選ばれた訓練サンプルの勾配を計算するが，SGD とは異なり，すべてのサンプルの勾配を保存し，それらを平均化することで新しい勾配を計算する．保存した勾配はパラメータ空間の現在地での新しい勾配によって置き換えられる．したがって，この手法によって計算された勾配によって，真の勾配を効率的に推定することができる．しかし，バッチ増加法とは異なり，SAG は，局所解のない強い凸性の損失関数の最適化に焦点を当てているので，SGD によって生み出された，真の誤差からの大きな違いという性質を利点として利用しない．

Stochastic Variance Reduction Gradient (SVRG) (Johnson & Zhang, 2013) もまた SGD の改良形である．ノイズ入りの勾配と真の勾配との間の違いの分散を減らす

表 4.1: 用いたモデル構造.BN はバッチ正規化を表す.

Layer type	Channels/Units
5×5 conv, BN, ReLU	32
2×2 max pool, str. 2	32
5×5 conv, BN, ReLU	64
2×2 max pool, str. 2	64
BN, ReLU	64
Dropout with $p = 0.5$	64
Fully connected, BN, ReLU	1024
Dropout with $p = 0.5$	1024
Output softmax	10

ことで，この手法は勾配の保存を要することなしに SAG と同程度の速さの収束を行える．しかし，SAG 同様，真の誤差からの大きな違いという性質を利用しない．

バッチサイズを徐々に増加させることは学習において効果的であると知られている．例えば，最近の研究 (Smith, Kindermans, & Le, 2017) において，Smith らは，学習中，学習率を減少させず，バッチサイズを増加させることが，学習を加速するのに役立つことを実証した．私たちの研究では，このことがシャープな局所解からの脱出とフラットな局所解への収束のためにも役立つかもしれないという点に注目している．

4.5　主要な実験

私たちは以下の実験によってバッチサイズ増加法の効果を調査した．実験では，畳み込みニューラルネットワーク (CNN) を MNIST データセット (LeCun et al., 1998) を用いて学習した．モデルには，表 4.1 に示す，比較的小さな CNN を用いた．ReLU(Glorot, Bordes, & Bengio, 2011) を隠れ層の活性化関数として用い，ソフトマックス関数を出力層の活性化関数として用いた．バッチ内の平均を 0 に分散を 1 に調整するバッチ正規化 (Ioffe & Szegedy, 2015) を，出力層を除いた各層の出力に適用した．過学習を抑制するために，ドロップアウト (Srivastava et al., 2014)

を全結合層に用い，ユニットを選択する確率はすべて 0.5 とした．Glorot の一様分布 (LeCun et al., 1998; Glorot & Bengio, 2010) をニューラルネットワークの重みの初期化に用いた．更新手法には，初期学習率を 0.1 とした AdaGrad を用いた．

小さいバッチサイズを用いることによる非定常な損失関数は収束を遅くするので，収束のために要求される更新数はバッチサイズに依存する．したがって，異なるバッチサイズのテスト誤差を比較するために，決められた回数だけ学習を行う代わりに，ほぼ収束したとみなされるまで学習を続けた．実際，3,000 エポックが収束のために十分だった．テスト誤差はエポックごとに計算し，訓練データは順序の影響を除くために毎エポックシャッフルした．

図 4.4 に示すように，バッチサイズ変更に関する 4 つの方法（線形増加 (a)→(d)，N_1 に固定 (a)→(c)，N_2 に固定 (b)→(d)，N_1 と N_2 の平均に固定 (e)→(f)）を用いてテスト誤差を比較することで，バッチサイズを変える効果を調査した．

効率上の理由から 10 より小さいバッチサイズは実用的にほとんど用いられないことを考慮して，N_1 は 10 に設定した．N_2 に関しては，2 つの異なる値（1 つめの条件は $N_2 = 100$，2 つめの条件は $N_2 = 500$）を設定した．これらの条件から固定したバッチサイズは 1 つめの条件では 10, 55, 100，2 つめの条件では 10, 255, 500 とした．線形増加の場合のバッチサイズ増加法のアルゴリズムをアルゴリズム 3 に示す．そこで，全体のサンプル数の範囲は 0 から $M-1$ までであり，$(i;i+N)$ は

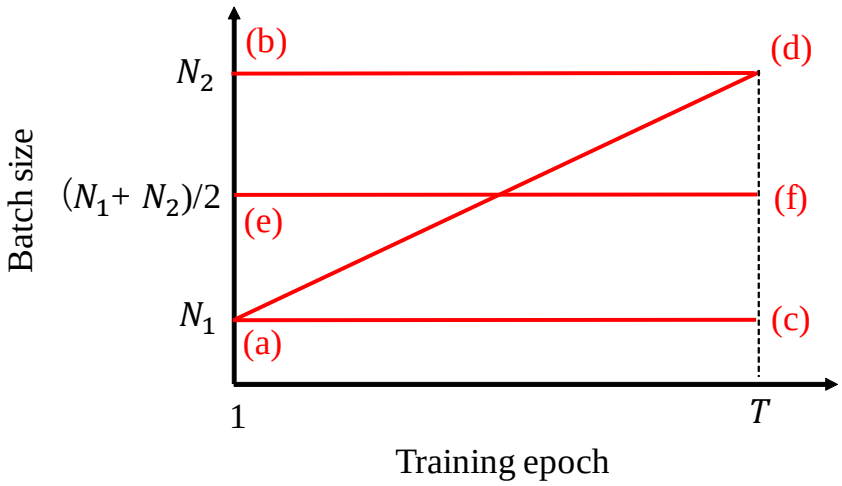


図 4.4: バッチサイズの変化方法.4 通りの方法 ((a)→(d), (a)→(c), (b)→(d), and (e)→(f)) が試された．

Algorithm 4 バッチサイズ増加法（線形増加）

Require: 全エポック数: $T(>1)$
初期バッチサイズ: N_1
最終エポックでのバッチサイズ: $N_2(>N_1)$
訓練データ数: M

```
for t from 0 to T − 1 do
  i ← 0
  N ← round((N2−N1)/(T−1) t + N1)
  while i < M do
    i ← min(i, M − N)
    θ ← θ − η · ∇θ J(θ; x^(i;i+N); y^(i;i+N))
    i ← i + N
  end while
end for
```

各ミニバッチ内のサンプル数の範囲である $[i,i+N)$ を表す．

すべてのサンプルを用いた交差エントロピーの平均であるテスト誤差の推移を図 4.5 に示す．その結果から，バッチサイズ増加法が最も低いテスト誤差と最も安定した推移を示したことがわかる．図 4.6 に示したテスト誤識別率の推移から，バッチサイズ増加法が最も低いテスト誤識別率を示したこともわかる．全体として，バッチサイズ増加法は最も良い結果を示した．

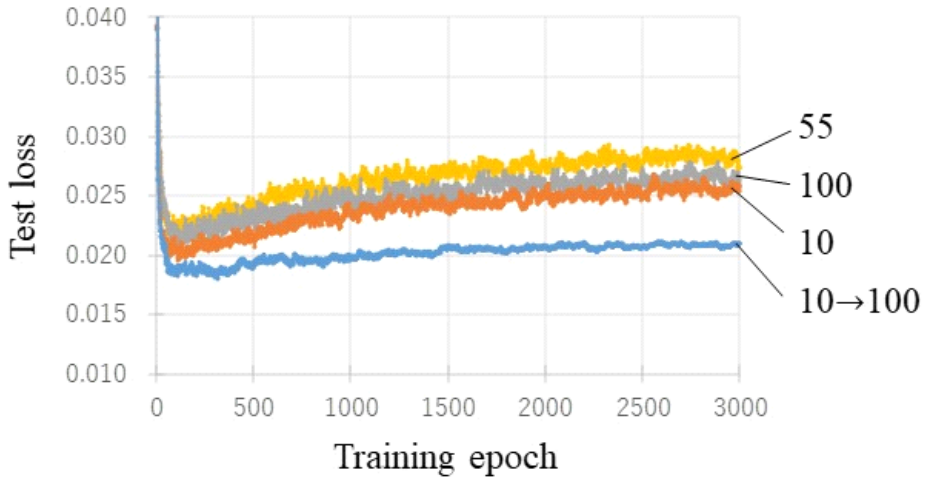
図 4.5 のテスト誤差の推移と図 4.7 に示した訓練誤差の推移から，固定したバッチサイズを用いた場合，過学習の強い傾向がみられることがわかる．一方で，バッチサイズ増加法を用いた場合，そのような傾向はみられなかった．バッチサイズ増加法を用いた場合，固定した大きいバッチサイズを用いた場合よりも，訓練誤差は小さくなかったのに対し，テスト誤差が小さかったため，パラメータはフラットな局所解に収束したと予測される．したがって，バッチサイズが徐々に増加することは，フラットな局所解への安定した収束のために効果的であるといえるだろう．

4.6　探索範囲の調査

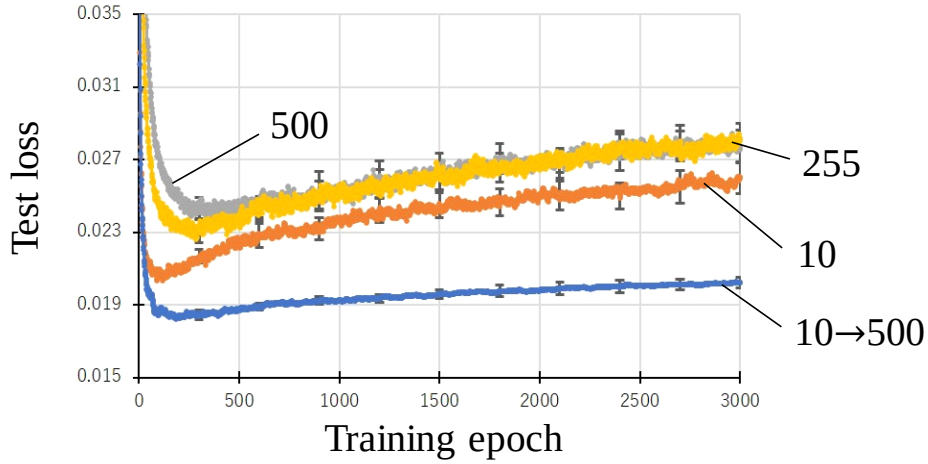
前節において，バッチサイズが小さいほど，探索範囲が広くなり，バッチサイズ増加法は前半に広い範囲を探索し，徐々に範囲を狭くするという予測を述べた．

本節では，探索範囲におけるバッチサイズのそのような効果を，実験を通して調査する．

ニューラルネットワークの重みパラメータ数を D とする．D 次元のパラメータ空間を多くのセルをもつメッシュに分割し，学習中に訪れたセルの数 N_{mesh} を数えたい．もし一つのセルを 2 回以上訪れたとしても，それを 1 回の訪問としてカウントする．しかし，もしすべての次元がセルに分割されるならば，私たちのニューラルネットワークにおいては D が 1,111,946 と，N_{mesh} が非常に大きくなってしま



(a) 最初の条件: $N_1 = 10$, $N_2 = 100$.

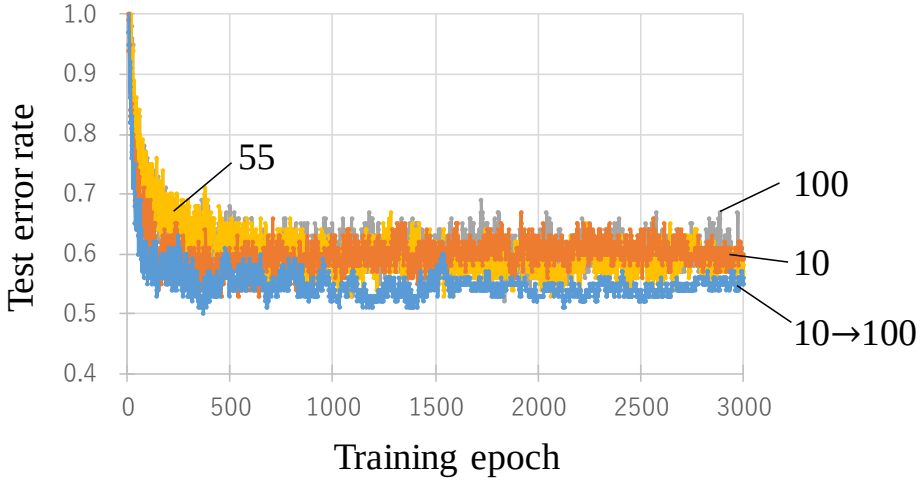


(b) 2 つめの条件: $N_1 = 10$, $N_2 = 500$.

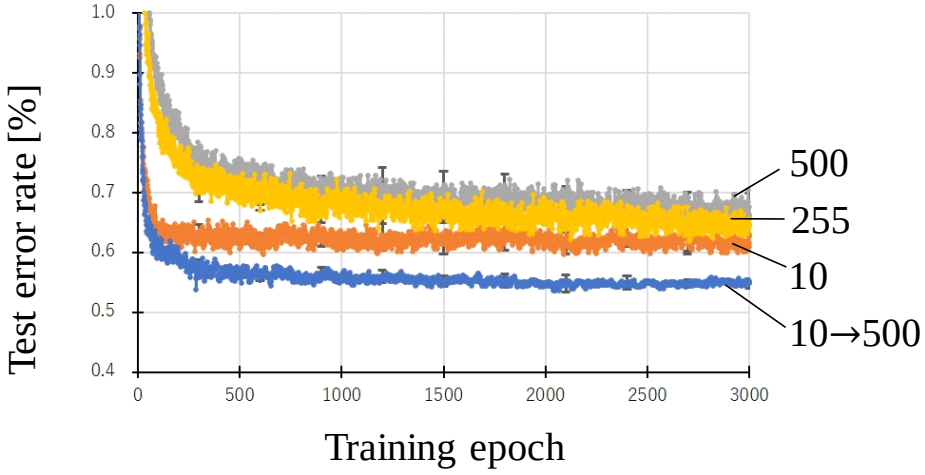
図 4.5: バッチサイズを変えることによるテスト誤差への影響.

う．その場合，学習中に同じセルをめったに訪問しなくなってしまい，異なったバッチサイズを用いたときの N_{mesh} の値を比較することが難しくなる．これを避けるために，D 次元からランダムに D' 次元を抽出し，それを等しい間隔で C 個のセクションに分割し，残りの $D - D'$ 次元は分割しない．結果として，$C^{D'}$ 個のセルをもったメッシュが得られる．本実験では，$D' = 1000$ に設定し，C が N_{mesh} に与える影響を調査するために，C の複数の値を用いた．

本実験では，パラメータの領域を，すべての次元について $[-0.3, 0.3]$ に固定し



(a) 最初の条件: $N_1 = 10$, $N_2 = 100$.



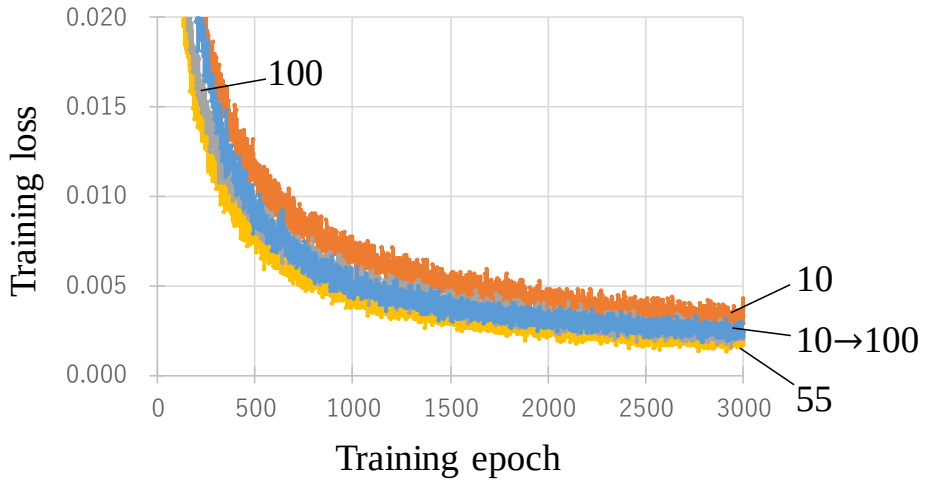
(b) 2 つめの条件: $N_1 = 10$, $N_2 = 500$.

図 4.6: バッチサイズを変えることによるテスト誤識別率への影響.

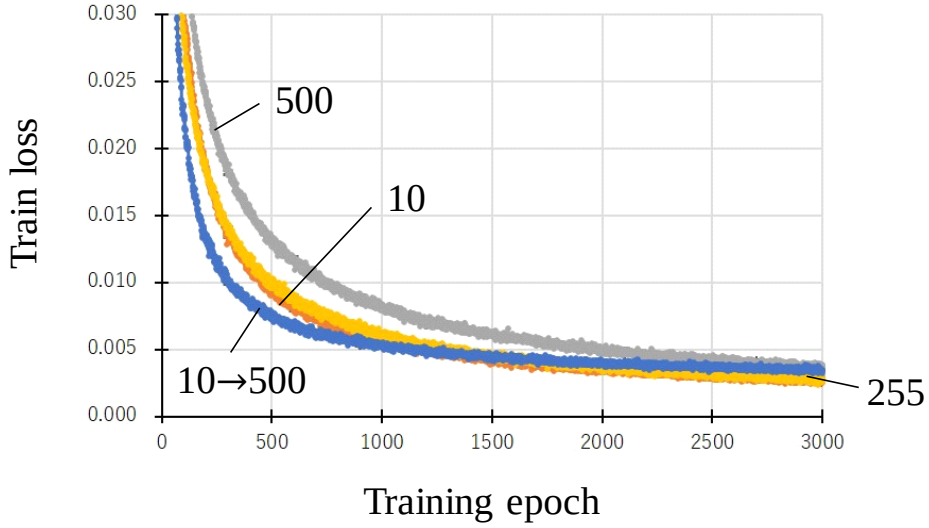
た．その範囲は，すべてのバッチサイズを用いた実験において，学習中にすべての重みがとった値が $[-0.23, 0.19]$ に収まったことから設定された．

前節で用いたのと同じモデルとデータを用い，図 4.4 の N_1, N_2, T をそれぞれ 10, 500, 1,000 に設定した．異なったバッチサイズに応じた結果を比較するために，各エポックでのパラメータの値をプロットし，全部で 1000 個のデータが得られた．

図 4.8 に示す実験結果から，バッチサイズ増加法を用いたときの N_{mesh} はバッチ



(a) 最初の条件: $N_1 = 10, N_2 = 100$.



(b) 2 つめの条件: $N_1 = 10, N_2 = 500$.

図 4.7: バッチサイズを変えることによる訓練誤差への影響.

サイズが 500 に固定されたときとほぼ同じであり，バッチサイズが 10 に固定されたときよりもずっと小さい．

しかし，過学習中の N_{mesh} の値と過学習前のその値を区別したい．それを調査するために，モデルが 1,000 エポック前には収束しているとみられることを考慮し，学習期間を，ある値 T_1 を境に前半と後半に分け，T_1 の前後の N_{mesh} を数えた．メッシュの構造に関して，$C \in \{10, 20, 30\}$ と設定した．T_1 の値は 100〜900 の範囲で変化させた．

図 4.9 にみられるように，バッチサイズ増加法は T_1 以降，最も小さい N_{mesh} の推移となった．したがって，バッチサイズ増加法はモデルが過学習し始めた後に，最も安定した収束を行うといえる．さらに，図 4.10 にみられるように，バッチサイズ増加法を用いたときの N_{mesh} は，エポックが $T_1 = 500$ に達する前，固定したバッチサイズが 500 のものより大きかった．

未学習時に，特に小さな訓練誤差をもった領域において，広い範囲を探索することは効率的な学習にとって重要である．なぜなら，局所解は一般に，初期状態での誤差と比較して非常に小さな誤差をもっており，学習の前半にそのような局所解から逃れることが良い汎化性能につながり得るからである．そこで，小さな誤差をもった領域において N_{mesh} を比較するために，あるセルにおける誤差 E がし

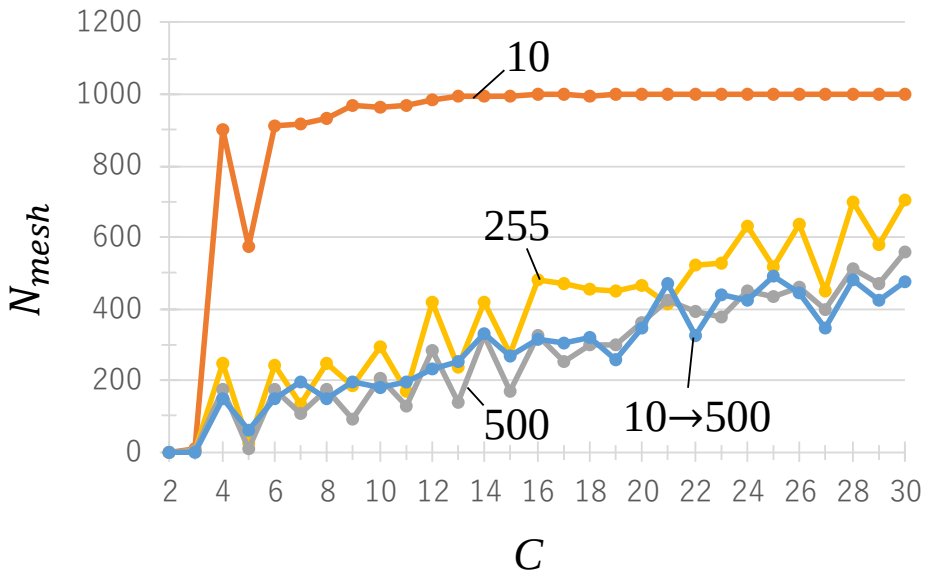


図 4.8: 学習全体で訪問したセル数の比較.

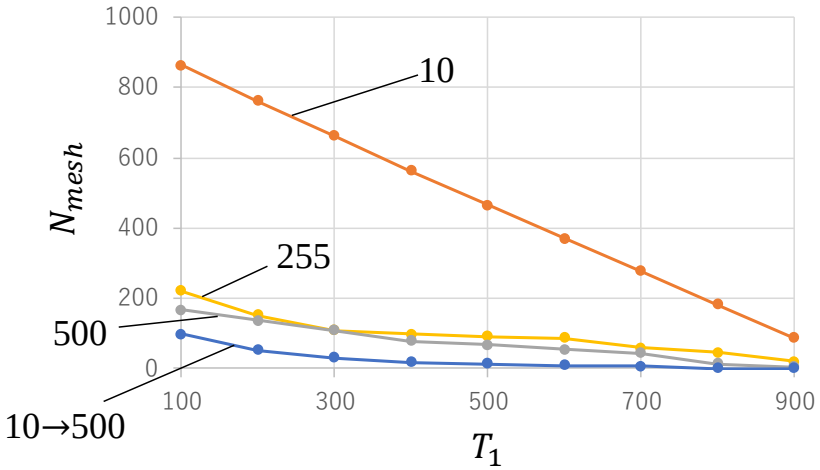
きい値 $E_{min} + E_1(E_{max} - E_{min})$ より小さいかどうかによって，メッシュを 2 つの領域に分ける．ここで，E_{max} と E_{min} は，それぞれメッシュにおける最大の誤差と最小の誤差である．E_1 はしきい値を決めるための 0 と 1 の間の値である．過学習の始まりのエポックの大まかな基準として $T_1 = 500$ とし，エポックが T_1 に達する前の $E < E_{min} + E_1(E_{max} - E_{min})$ である領域における N_{mesh} をカウントした．

図 4.11 にみられるように，$C = 20, 30$ および $E_1 > 0.1$ のとき，バッチサイズ増加法を用いたときの N_{mesh} は 255 や 500 にバッチサイズを固定したときよりも大きかった．これは,（細かいメッシュにするために）C が十分大きく,（過学習前に訪れた狭い領域を評価するために）E もまた十分大きいとき，バッチサイズ増加法が，大きな値に固定されたバッチサイズを用いたときよりも広い領域を探索することができたことを意味する．

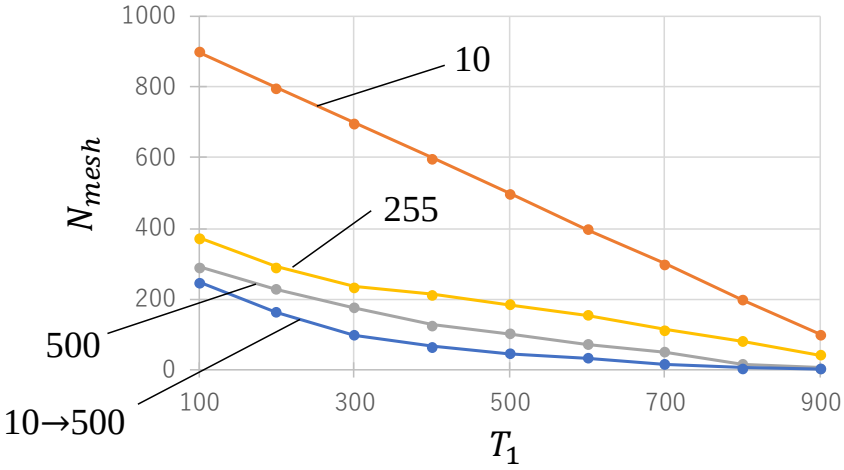
4.7　まとめ

本節では，非凸最適化問題を解くために，シミュレーテッドアニーリングのような効果的な探索手法の包括的な枠組みを与えた．繰り返される探索ステップの方向に加えられたノイズが，ノイズが非常に小さいときそれらがほぼ同じであるということを除き，この枠組みを標準的な勾配降下法と異なるものにする．この枠組みに基づいた探索手法がなぜ悪い局所解を避けることができるのかについて論じた．

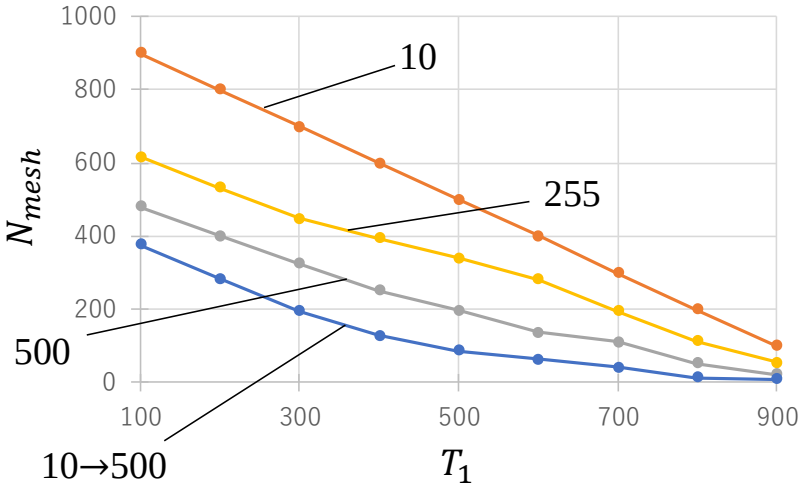
この枠組みを，ニューラルネットワークのミニバッチ学習へ適用することで，大きなバッチサイズはより安定した損失関数を生み出す傾向にあることを示した．そして，それはなぜ大きなバッチサイズがしばしば汎化性能を悪化させるかを説明している．この枠組みとこの議論に基づいて，ニューラルネットワークの非凸最適化問題を解くためのバッチサイズ増加法を開発した．そしてこの戦略が汎化性能の改善につながり得ることを実験的に示した．



(a) $C = 10$.

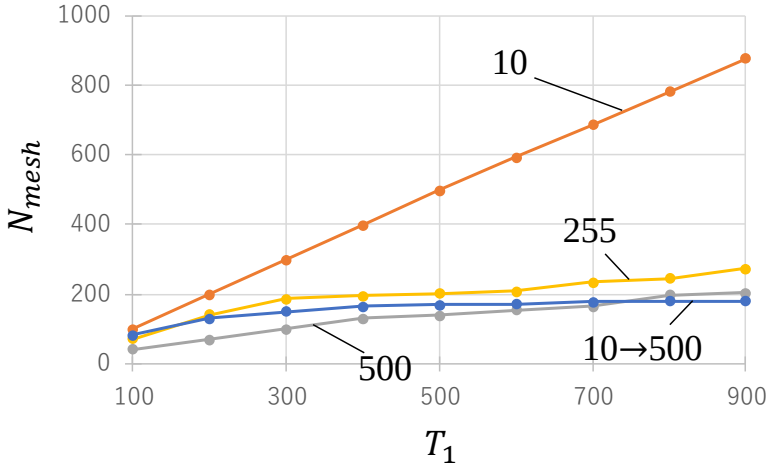


(b) $C = 20$.

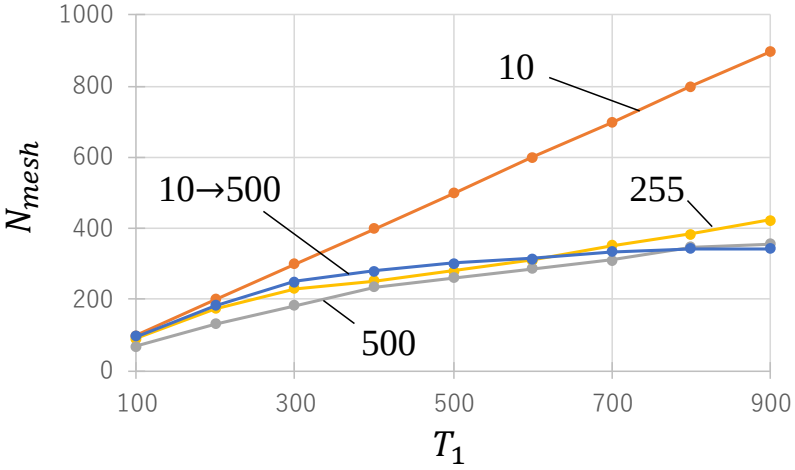


(c) $C = 30$.

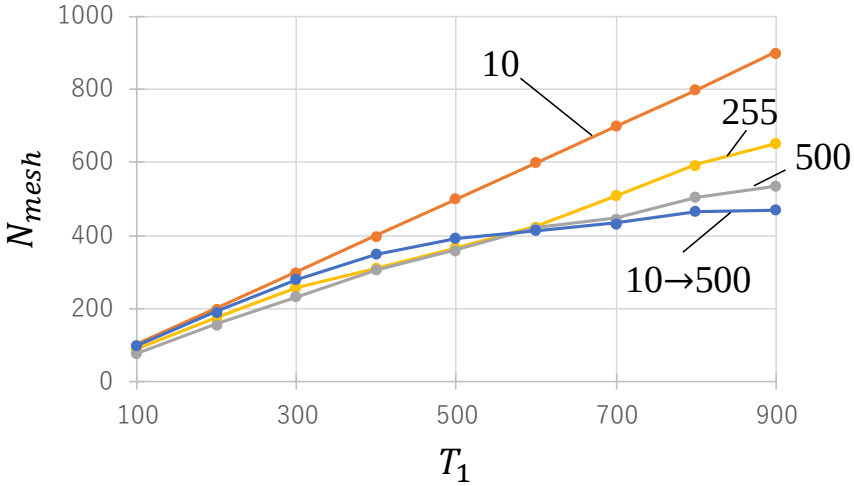
図 4.9: 学習の後半において訪問したセル数の比較.



(a) $C = 10$.

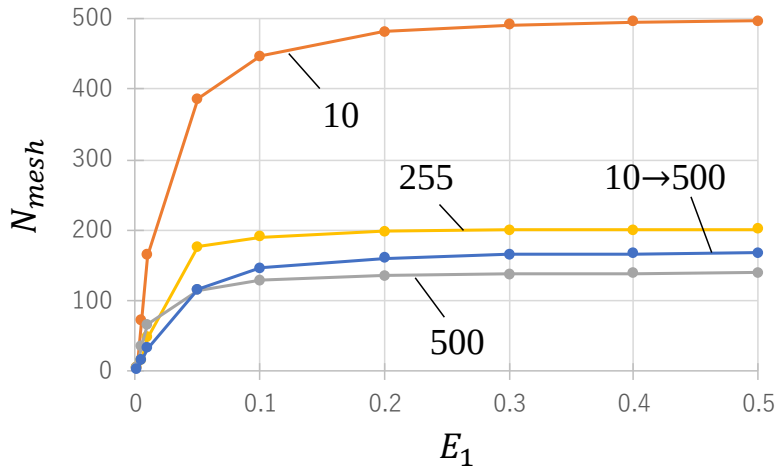


(b) $C = 20$.

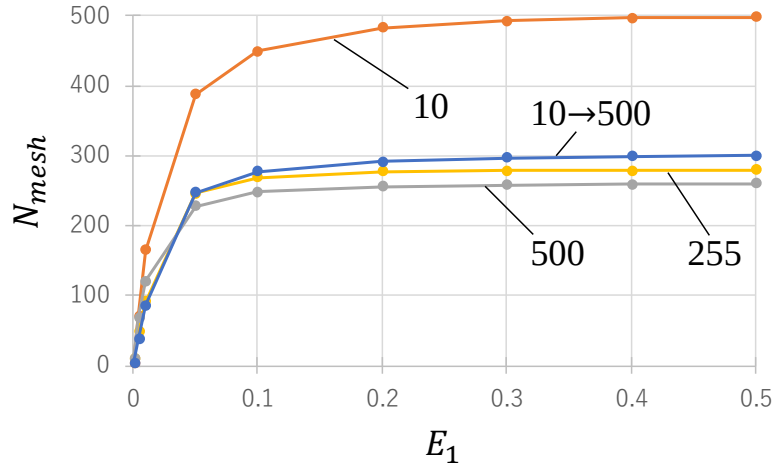


(c) $C = 30$.

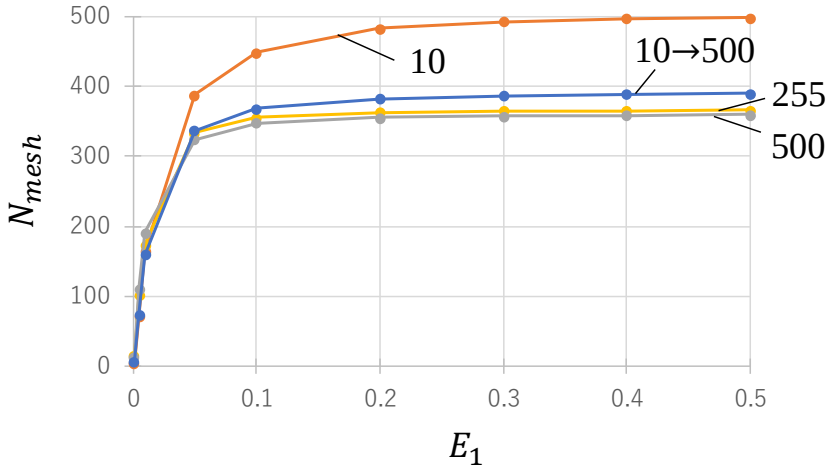
図 4.10: 学習の前半において訪問したセル数の比較.



(a) $C = 10$.



(b) $C = 20$.



(c) $C = 30$.

図 4.11: $T_1 = 500$ 以前での $E < E_1$ の領域において訪問したセル数の比較.

第5章　結論

本研究では，ニューラルネットワークの損失関数における解の効果的な探索について研究を行った．損失関数は多くの凹凸を含んだ非凸な関数となるが，単純な勾配降下法を用いると局所解にトラップされやすい．誤差の大きな局所解にトラップされるならば，低いモデル精度が得られてしまう．この問題を解決するために，学習の前半は広く探索を行い，後半は安定した収束を行うという学習戦略を考え，それをニューラルネットワークに適用するために，学習率およびバッチサイズを制御することにより効果的な学習を行う ALR 法およびバッチサイズ増加法を提案した．

ALR 法では，エポックごとに複数の学習率を用いて学習を試み，訓練誤差が最も小さくなるときの学習率を選択する．学習中学習率がどのように変化するかは，損失関数の形状に依存するが，もし学習の前半は学習率が増加し，後半は減少するという変化が起こるならば，学習の前半は広い探索を行い，後半は安定した収束を行えることになり，学習戦略を反映した学習となる．また，ALR 法は分岐数，スケールファクター，ビームサイズという 3 つの主要パラメータを含み，これらの設定が結果に大きな影響を与える．分岐数およびビームサイズは大きい方が広い範囲を探索できるのでより良い解を見つけやすくなるが，計算時間や状態の記憶量が増大するので，それらの影響を考慮してパラメータを設定する必要がある．スケールファクターは計算時間や記憶量に影響を与えないが，その設定は他のパラメータよりも難しい．学習の前半において，学習率の増加の回数の割合を考える．その割合が 0 に近ければ学習率は減少するか変化しないため，悪い局所解にとらわれやすいと考えられる．学習の前半における学習率の増加の回数の割合が程良くなるようなスケールファクターの設定が良い学習のために重要であると考え，実験によっても実証した．

ALR 法に関する実験では，MNIST および UCI 機械学習リポジトリ (Car Evaluation, Wine, Letter Recognition) データセットを用いて 3 層ニューラルネットワーク

を学習し，ALR 法が従来の手法に比べて訓練誤差が小さく，またテスト精度も向上することが確かめられた．そのときの学習中の学習率は，序盤に増加し，その後減少するという推移となったので，学習戦略に合った変化をとったといえる．現状では，畳み込みニューラルネットワークに ALR 法を適用すると学習に失敗するので，その点を改善することが今後の課題である．

バッチサイズ増加法は，学習中ミニバッチのサイズを線形に増加させるという，実装の容易な手法である．この手法は他の研究にもみられるが，本研究は探索の制御と結び付けた点が新しい．バッチサイズ増加法は学習中バッチサイズを変化させることで損失関数の非定常的性質を制御する．学習の前半は小さなバッチサイズを用い関数を非定常的にすることで，局所解を避けて広い範囲を探索できるようにし，学習の後半は大きなバッチサイズを用い関数を定常的にすることで，安定した収束を行えるようにするという狙いである．この考え方はシミュレーテッド・アニーリングと同じ枠組みで抽象化することができ，学習戦略をニューラルネットワーク学習へと適用することができたと捉えることができる．

バッチサイズ増加法に関する実験では，MNIST データを用いて畳み込みニューラルネットワークを学習し，バッチサイズを固定した場合と比べ，バッチサイズ増加法は過学習を抑制でき，優れたテスト精度を示した．バッチサイズ増加法を用いることで実際に探索範囲が制御できていることを示すために，同じ実験条件のもと，細かいメッシュをもったパラメータ空間において，学習中に訪問したセル数を比較した．バッチサイズ増加法は，バッチサイズを固定した場合と比べ，学習の序盤は誤差の小さい領域において広い範囲を探索し，終盤は狭い範囲を探索するという期待した結果が得られた．

謝辞

本研究を行うにあたり，的確なご助言，ご指導を頂きました知能ソフトウェア研究室の栗原正仁教授と小山聡准教授に深く感謝いたします．また，副指導教員としてご指導頂きました自律系工学研究室の山本雅人教授，調和系工学研究室の川村秀憲教授，ヒューマンコンピュータインタラクション研究室の小野哲雄教授にも，心より感謝申し上げます．また，知能ソフトウェア研究室の先輩後輩の方々にも，感謝申し上げます．

参考文献

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. *New York: Oxford Univ. Press*.

Breuel, T. M. (2015). On the Convergence of SGD Training of Neural Networks. *ArXiv Preprint arXiv:1508.02790*.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*, 41 – 51.

Daniel, C., Taylor, J., & Nowozin, S. (2016). Learning Step Size Controllers for Robust Neural Network Training. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*.

Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp Minima Can Generalize For Deep Nets. *ArXiv Preprint arXiv: 1703.04933*.

Duchi, J., Hazen, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research (JMLR)*, *12*, 2121 – 2159.

Egmont-Petersen M., Talmon, J. L., Hasman, A., & Ambergen, A. W. (1998). Assessing the importance of features for multi-layer perceptrons. *Neural Networks*, *11(4)*, 623 - 635.

Fayek, H. M., Lech, M., & Cavedon, L. (2017). Evaluating deep learning architectures for Speech Emotion Recognition. *Neural Networks*, *92*, 60 - 68.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of Artificial Intelligence and Statistics Conference (AISTATS)*, *9*, 249 – 256.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. In *Proceedings of Artificial Intelligence and Statistics Conference (AISTATS)*, *15*, 315 – 323.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, *13*, 533 – 549.

Hardt, M., Recht, B., & Singer, Y. (2015). Train faster, generalize better: Stability of stochastic gradient descent. *ArXiv Preprint arXiv:1509.01240*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29(6)*, 82 - 97.

Huang, Z., Zweig, G., Levit, M., Dumoulin, B., Oguz, B., & Chang, S. (2013). Accelerating recurrent neural network training via two stage classes and parallelization. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, *37*, 448 – 456.

Johnson, M. W., Amin, M. H. S., et al. (2011). Quantum annealing with manufactured spins. *Nature*, *473*, 194 – 198.

Johnson, R. & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*.

Kadowaki, T. & Nishimori, H. (1998). Quantum Annealing in the Transverse Ising Model. *Physical Review E*, *58*, 5355 – 5363.

Kanada, Y. (2016). Optimizing neural-network learning rate by using a genetic algorithm with per-epoch mutations. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2016)*.

Kawaguchi, K. (2016). Deep Learning without Poor Local Minima. In *Advances in Neural Information Processing Systems (NIPS)*.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Kingma, D. & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*, 671 – 680.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K-R. (1998). Efficient BackProp. *Neural Networks: Tricks of the Trade*, 9 – 48.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE, 86(11)*, 2278 - 2324.

Lichman, M. (2013). UCI Machine Learning Repository *http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science*.

Lu, H. & Kawaguchi, K. (2017). Depth Creates No Bad Local Minima. *ArXiv Preprint arXiv:1702.08580*.

Lundy, M. & Mees, A. (1986). Convergence of an Annealing Algorithm. *Programming*, *34*, 111 – 124.

Mandt, S., Hoffman, M. D., & Blei, D. M. (2016). A Variational Analysis of Stochastic Gradient Algorithms. *ArXiv Preprint arXiv:1602.02666*.

Mishkin, D. & Matas, J. (2015). All you need is a good init. *ArXiv Preprint arXiv:1511.06422*.

Neyshabur, B., Salakhutdinov, R., & Srebro N. (2015). Path-SGD: Path-Normalized Optimization in Deep Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Park, J. G. & Jo, S. (2016). Approximate Bayesian MLP regularization for regression in the presence of noise. *Neural Networks*, *83*, 75 - 85.

Rere, L. M. R., Fanany, M. I., & Arymurthy, A. M. (2015). Simulated Annealing Algorithm for Deep Learning. *Procedia Computer Science*, *72*, 137 – 144.

Roux, N. L., Schmidt, M., & Bach, F. R. (2012). A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets. In *Advances in Neural Information Processing Systems (NIPS)*.

Sainath, T. N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A. R., Dahi, G., & Ramabhadran, B. (2015). Deep Convolutional Neural Networks for Large-scale Speech Tasks. *Neural Networks*, *64*, 39 – 48.

Shi, Z., Ye, Y., & Wu, Y. (2016). Rank-based pooling for deep convolutional neural networks. *Neural Networks*, *83*, 21 – 31.

Smith, S. L., Kindermans, P. J., & Le, Q. V. (2017). Don't Decay the Learning Rate, Increase the Batch Size. *ArXiv Preprint arXiv:1711.00489*.

Song, Y., Schwing A. G, Zemel R. S., & Urtasun R. (2016). Training Deep Neural Networks via Direct Loss Minimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*.

Srebro, N. & Shraibman, A. (2005). Rank, Trace-Norm and Max-Norm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT 2005)*, 545 - 560.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, *15*, 1929 – 1958.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 1139 – 1147.

Sutton, R. & Barto, A. (1998). Reinforcement learning: An introduction. *MIT Press Cambridge*.

Tang, J., Deng, C., & Huang, G.-B. (2015). Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, *27(4)*, 809 - 821.

Tieleman, T. & Hinton, G. E. (2012). Lecture 6.5 - RMSProp *COURSERA: Neural networks for machine learning*.

Wu, H. & Gu, X. (2015). Towards dropout training for convolutional neural networks. *Neural Networks*, *71*, 1 - 10.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *ArXiv Preprint arXiv:1212.5701*.

研究業績

学術雑誌

Tomoumi Takase, Satoshi Oyama, and Masahito Kurihara: Why does Large Batch Training Result in Poor Generalization?—A Comprehensive Explanation and a Better Strategy from the Viewpoint of Stochastic Optimization, *Neural Computation,* Vol. 30(7), pp. 2005-2023, July 2018.

Tomoumi Takase, Satoshi Oyama, and Masahito Kurihara: Effective Neural Network Training with Adaptive Learning Rate Based on Training Loss, *Neural Networks,* Vol. 101, pp. 68-78, May 2018.

国際会議

Tomoumi Takase, Satoshi Oyama, and Masahito Kurihara: Longer Distance Weight Prediction for Faster Training of Neural Networks, *Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC2018),* October, 2018 (掲載決定,10 月公表予定).

国内会議

高瀬 朝海, 小山 聡, 栗原 正仁, ニューラルネットワーク学習における訓練誤差に基づいた学習率調整法, 情報処理北海道シンポジウム 2017 , 札幌, 10 月・2017 年.