目次

[学習の大まかな流れ 2](#_Toc164613000)

[活性化関数 3](#_Toc164613001)

[隠れ層 3](#_Toc164613002)

[出力層 3](#_Toc164613003)

[損失関数 3](#_Toc164613004)

[重みパラメータの初期値の決め方(手法) 4](#_Toc164613005)

[誤差逆伝播法 4](#_Toc164613006)

[勾配降下法 4](#_Toc164613007)

[確率的勾配降下法 4](#_Toc164613008)

[モーメンタム 4](#_Toc164613009)

[AdaGrad 5](#_Toc164613010)

[Adam 5](#_Toc164613011)

[Batch Normalization 5](#_Toc164613012)

[.過学習の抑制 6](#_Toc164613013)

[ハイパーパラメータの最適化 6](#_Toc164613014)

## 学習の大まかな流れ

|  |  |
| --- | --- |
| STEP1データをバッチに分ける | 60000個のデータを100個のデータで  1バッチとして、600バッチ用意 |
| STEP2 バッチをランダムに選択 | 600バッチから1バッチをランダムに選択する |
| STEP3　前向き伝播  ※1バッチ目でパラメータの初期値を決めるため、XavierやHeなどの手法が使われる  ※誤差逆伝播法で学習効率を上げるため、正規化を行うためBatch Normalizationなどを使う | 選択したミニバッチのデータをモデルに入力して、出力を得る |
| STEP4損失の計算 | モデルの出力である予測値と実際のラベルの正解データを比較して損失関数を求める |
| STEP5 誤差逆伝播法  ※100個のデータが含まれるミニバッチに対して損失関数の平均を一度計算した後、その平均損失に基づいて一度だけ勾配を計算し、パラメータ（重みとバイアス）を更新します | 計算された損失関数から後ろ向き伝播により勾配を求めて、モデルの各パラメータに関して損失を最小化する方向を計算 |
| STEP6 パラメータの更新  ※更新の仕方は手法に応じて違う | 求められた勾配方向に、あらかじめ設定した学習率に応じて、パラメータ（重みとバイアス）を更新 |

・ミニバッチ全てに対して繰り返すか、満足いくまで行う

　2バッチ目では、1バッチ目で更新した重みのパラメータからスタートする

　3バッチ目では、2バッチ目で更新した重みのパラメータからスタートする

　(これを繰り返す)

※600バッチの場合は、600回誤差逆伝播法が行われる

## 活性化関数

順方向伝播において、前の層からの入力に対してアフィン変換（線形変換：重み付き和＋バイアス）を行い、それを基に次の層への出力（または最終出力）を決定

### 隠れ層

シグモイド関数→出力を確率として解釈することができるため、二値分類問題の出力層によく用いられる

ReLu関数→多くの問題において良好な結果をもたらす。特に、勾配消失問題の影響を受けにくいため、深いネットワークにおいて有効

### 出力層

　　・ソフトマックス関数　→　分類問題

・恒等関数　→　回帰問題

## 損失関数

交差エントロピー誤差→分類問題で有効

二乗和誤差→回帰問題で有効

## 重みパラメータの初期値の決め方(手法)

シグモイド関数→Xavier

ReLu関数→He

## 誤差逆伝播法

### 勾配降下法

変化率が負の値なら、重りの値を増やすと局所的な最小値に近づく

変化率が正の値なら、重りの値を減らすと局所的な最小値に近づく

### 確率的勾配降下法

比較的単純な問題や大規模なデータセットに適している

### モーメンタム

パラメータの更新を滑らかにしたもの

前回の更新量を考慮することで、勾配が小さい領域を効果的に通過し、収束を早めることができる。

現在の勾配と、前回の更新量に一定の割合を乗した値を組み合わせる

パラメータを更新していっても、更新されず同じ値をいききしたり、損失関数が複雑な場合に有効

### AdaGrad

各パラメータに対して個別の学習率を適用することで、学習過程を改善

全てのパラメータが同じ量だけ更新されるのではなく、頻繁に更新されるパラメータは小さく、稀にしか更新されないパラメータは大きく更新されるように、学習率を自動的に調整

データセット内のデータが多くの次元を持つが、それらの大部分の要素がゼロの場合に有効

### Adam

*モーメンタムとAdaGradの融合*

様々なタイプの問題に対して良好な性能を示し、特にディープラーニングにおいて広く使われてる多くの場合デフォルトで設定

## Batch Normalization

ニューラルネットワークの学習を安定させ、加速させるための手法

順方向において、アフィン変換の後、活性化関数にデータが渡る前に正規化する

## .過学習の抑制

Weight decay

重みの値が大きすぎると、過学習が起こってしまうので、重みの絶対値を小さくすることで過学習を防ぐ

損失関数に重みの二乗の和を加えることで実現

Dropout

ニューラルネットワークのモデルが複雑の場合に有効

訓練時、隠れ層の各層にあるニューロンをランダムに選択して、一定の確率でそのニューロンを無効化する。

選択されたニューロンはその学習のステップでの活動を行わなくなる

## ハイパーパラメータの最適化

ハイパーパラメータとは各層のニューロンの数、バッチサイズ、パラメータの更新の際の学習係数やWeight decayなど

ハイパーパラメータは検証データを使用して評価する

訓練データ、テストデータ、検証データの3つを用意する必要がある

手順１：ハイパーパラメータの範囲を設定(0.001から1000)

手順２：設定したハイパーパラメータの範囲から、ランダムにサンプリングする

手順３：サンプリングされたハイパーパラメータの値を使用して学習を行い、検証データで認識精度の評価を行う

手順４：手順2と3を決めた回数繰り返して、それらの認識精度の結果から、ハイパーパラメータの範囲を狭める