深層学習 day2

Section1: 勾配消失問題

- 誤差逆伝播法が下位層に進むにつれ勾配がどんどん緩やかになっていく
- 勾配降下法による更新では下位層のパラメータがほとんど変わらず学習が失敗
- シグモイド関数は出力0~1を取るが、大きな値でも出力の変化が微小なため勾配消失問題を引き起こす

確認テスト1

シグモイド関数を微分した時、入力値0で最大値を取るがその値は？
答え：0.25

勾配消失問題の解決方法

- 活性化関数の選択をsigmoidではなくReLUにする
  - 勾配消失問題の回避
  - スパース化（不要部分がそぎ落とされる）
- 重みの初期値設定
  - Xavier（ザビエル）：重みの要素を前の層のノード数の平方根で除算した値
  - XavierはReLU、sigmoid, 双曲線正接関数
  - He: Xavierに√2をかけ合わせた値。He初期値を設定する際の活性化関数はReLU関数。

確認テスト2

重みの初期値に0を設定した時の問題点
答え：正しい学習が行えない。全ての重みの値が均一に更新されるため、多数の重みを持つ意味がなくなる。

バッチ正規化

ミニバッチ単位で入力値のデータの偏りを抑制。

確認テスト3:

バッチ正規化の降下を2点挙げよ。
答え：学習が速く進む。過学習を抑制。

実装演習結果

ファイル：my_2_2_1_vanishing_gradient.ipynb

- 重みを全て0に設定+sigmoid

```
Generation: 2000. 正答率(トレーニング) = 0.13
                : 2000. 正答率(テスト) = 0.1135
```

- 重みの初期値ランダム+sigmoid

```
Generation: 2000. 正答率(トレーニング) = 0.15
                : 2000. 正答率(テスト) = 0.101
```

- 重みの初期値ランダム＋ReLU

```
Generation: 2000. 正答率(トレーニング) = 0.91
                : 2000. 正答率(テスト) = 0.9168
```

- 重みの初期値0+ReLU

```
Generation: 2000. 正答率(トレーニング) = 0.07
                : 2000. 正答率(テスト) = 0.1135
```

- 重みの初期値Xavier + sigmoid

```
Generation: 2000. 正答率(トレーニング) = 0.77
                : 2000. 正答率(テスト) = 0.7956
```

- 重みの初期値He + ReLU

```
Generation: 2000. 正答率(トレーニング) = 0.98
                : 2000. 正答率(テスト) = 0.9476
```

- 重みの初期値He + sigmoid(お試し)

```
Generation: 2000. 正答率(トレーニング) = 0.8
                : 2000. 正答率(テスト) = 0.8265
```

Section2: 学習率最適化手法

optimizerとして以下のような手法がある。

- モメンタム
- AdaGrad
- RMSProp
- Adam
    - 最近はこれが主流。モメンタムとRMSPropのいいとこどり。

確認テスト1: モメンタム、AdaGrad、RMSPropの特徴は？

- モメンタム
    - 過去の勾配の指数関数的減衰平均
    - 局所的最適解にならず大域的最適解となる
    - 谷間についてから最も低い値に速くたどり着く
    - なかなか値が収束しない
- AdaGrad
    - 勾配の緩やかな斜面に対して最適値に近づける
    - 学習率が徐々に小さくなるため鞍点問題を引き起こすことがある
- RMSProp：AdaGradの進化形。
    - 過去の勾配の2乗の指数関数的減衰平均
    - 局所的最適解にならず大域的最適解となる
    - ハイパーパラメータの調整が必要な場合が少ない

実装演習

ファイル：my_2_4_optimizer.ipynb
以下の課題を実施した。

- [try]学習率を変える
- [try]活性化関数と重みの初期化方法を変える
- [try]バッチ正規化をしてみる

Section3: 過学習

- 過学習：テスト誤差と訓練誤差とで学習曲線が乖離すること
- 特定の訓練データに特化して学習した結果、汎化能力が落ちる
- 原因はパラメータの数が多い、パラメータの値が適切でない、ノードが多いなど。

    - ネットワークの自由度が高い

正則化

正則化：ネットワークの自由度を制約

- L1, L2正則化
- ドロップアウト

Weight decay（荷重減衰）

- 重みが大きい値を取ることで過学習が発生することがある
- 誤差に対して正則化項を加算することで重みを抑制
    - 極端に大きい重みを取らないようにするための仕組み

L1/L2正則化

誤差関数にpノルムを加えたもの。ノルムは距離のこと。

種類	計算式	特徴
L1正則化（ラッソ回帰）	誤差関数にp=1ノルムを加算	スパース推定
L2正則化（リッジ回帰）	誤差関数にp=2ノルムを加算	縮小推定

確認テスト

L1正則化を示すグラフはどちらか？
答え：右のLassoがL1正則化である。

例題チャレンジ

- L2パラメータ正則化の問題
    - 答え：(4)param
- L1パラメータ正則化の問題
    - 答え：(3)np.sign(param)

ドロップアウト

- ランダムにノードを削除して学習させること
- データ量は変わらないが、異なるモデルを学習させているような扱いとなる
- 学習時間が伸びる

実装演習

ファイル：my_2_5_overfiting.ipynb
— [try]weight_decay_lambdaの値を変更して正則化の強さを確認

- weight_decay_lambdaの値を0.5に変えたところ学習が進まなかった。
- [try] dropout_ratioの値を変更
  - dropout_ratioを0.5とすると学習が進まなかった
- [try] optimizerとdropout_ratioの値を変更
  - optimizerをAdamにdropout_ratioを色々変更させたが、上手く学習されなかった

Section4: 畳み込みニューラルネットワークの概念

- CNN: 畳み込みニューラルネットワーク
- CNNの構造
  - 入力層→畳み込み層→畳み込み層→プーリング層→畳み込み層→畳み込み層→プーリング→全結合→出力層
- 畳み込み層
  - 3次元のデータをそのまま学習し次に伝えることが可能。
  - 入力画像に対し、フィルターを掛け合わせて足しこんだ値が出力値となる
  - パディング、ストライド、チャンネルといったパラメータがある
    - パディング：画像の周囲に指定したサイズだけ値を詰める。
    - ストライド：フィルターを適用時の移動サイズ
    - チャンネル：RGBなど。
- プーリング層
  - 対象領域のMax値もしくは平均値を取得
  - プーリングによりデータサイズを削減

確認テスト

サイズ6x6の入力画像をサイズ2x2のフィルターで畳みこんだ時の
出力画像のサイズを答えよ。
なお、ストライドとパディングは1とする。
答え：((6-2+1+1)/1)+1=7　よって出力サイズ=7

実装演習

ファイル：my_2_6_simple_convolution_network.ipynb
以下の演習を実施。

- [try] im2colの処理を確認しよう
- [try] col2imの処理を確認しよう

Section5: 最新のCNN

- AlexNet
  - 入力サイズ224x224を11x11フィルタで畳み込み
  - サイズ4096の全結合層の出力にDropoutを使用し過学習を防止している

その他CNNモデル

- GoogleNet: 2014年のILSVRC優勝モデル。
  - Inceptionモジュールと呼ばれる小さなネットワークを繋げて1つのモデルを作っている
  - Inceptionモジュールは複数のConvolution, poolingで構成。
- VGGNet: 2014年のILSVRCの2位モデル。ネットワークを深くし、認識精度を改善。
- ResNet: 2015年のILSVRC優勝モデル。ネットワークを深くした際の学習効率を改善したモデル。
  - residualモジュール
  - Batch Normalization：バッチ正規化。Resnet以降のモデルでは標準的に使用。
- MobileNet: モバイル機器用軽量版のCNN