Deep Learning 輪読会 2017

第11章　実用的な方法論

2017.12.04

松尾研究室
M1 阿久澤圭

構成

11.1 性能指標

11.2 初期のベースラインモデル

11.3 データの追加収集の判断

11.4 ハイパーパラメータの選択

11.5 デバッグの戦略

11.6 例：複数桁の数字認識

はじめに

- 前章までの議論：
  - 畳み込み、回帰結合など、特定のタスクに特化した手法の紹介
- 本章の議論：
  - アルゴリズムを正しく適応する実用的な方法論
    - 優秀な機械学習の専門家：機械学習の手法に精通 **＋ 実用的な方法論への理解**

- 実用的な方法論とは（例）
  - データをどれくらい集めるか
  - モデルの容量をどうするか
  - 正則化を使うか
  - モデルの実装のデバッグをどうするか
  - etc

はじめに

- 本章では以下の実用的な設計工程を推奨する
  - 目標の決定: 性能指標およびその目標値の決定
  - 全工程の確立
  - ボトルネックの見極め
  - ボトルネックの修正

11.1 性能指標

- 最初にすべきこと: どの性能指標で、どれくらいの目標値を目指すかの決定

  - 性能**指標**の選び方:
    - 一般的に、モデルの学習に用いられるコスト関数ではなく、正解率(誤差率)
    - タスクによっては、適合率、再現率、F値、網羅率など、高度な指標を使う
  - 性能**水準**の選び方
    - 学術: 過去のベンチマーク
    - 現実: アプリの安全性、費用対効果など様々な事情を加味する

11.2 初期のベースラインモデル

- 11.1で性能指標と目標が決まったら、動作するベースラインシステムを組む
- 考慮すべき事柄
  1. 対象とする問題の複雑さ
     - 簡単なタスクなら簡単なモデル（ロジスティック回帰など）
     - 「AI-complete」(画像、音声、NLPなど)なら深層学習
  2. 対象とするデータ構造
     - トポロジー構造を持つ => CNN
     - 入力または出力が系列 => RNN
  3. 最適化アルゴリズムの選択
     - SGD with scheduling or Adam
     - バッチ正規化は早い段階で試す価値あり（特に畳み込み、シグモイドに効く?）

11.2 初期のベースラインモデル

- 考慮すべき事柄（続き）
  4. 正則化
     - 訓練データが数千万を超えない限り、正則化を含めた方が良い
     - 早期終了はほぼ普遍的に使用されるべきである
       - 最近は反論もある[Soudry+ 2017]
         - » https://openreview.net/forum?id=r1q7n9gAb&noteId=HkS9oWtef
     - ドロップアウトは実装が簡単で効果が高い
  5. 既存研究のサーベイ
     - 類似タスクを解いている研究があれば、モデル構造や訓練済みモデルを借りてくる
  6. 教師なし学習（教師なし事前学習、半教師学習）を使うか
     - 教師なし学習が重要であるとわかっている分野では使うべき
       - NLPでは教師なし単語埋め込み学習が有効
       - CVではラベルが極端に少ない時の半教師学習が有効[Kingma+ (2014)]

11.3 データの追加収集の判断

- アルゴリズムを改善するよりも、データを増やす方が性能指標改善効果が大きい場合が多い

- データの追加収集をするかの判断材料
  - 訓練集合での性能が十分か
    - 隠れ層の追加・ハイパラ調整の限界効果が大きい時は、データ追加は見送る
    - モデルがうまく機能しない時、データの品質に原因があることも
  - 訓練集合での性能が十分なとき、テスト集合での性能は十分か
    - テスト集合で性能が落ちる（汎化誤差が大きい）とき、データ追加収集の効果が期待できる
    - ただし、データの追加収集と他の正則化手法の費用対効果を比べる必要あり
      - e.g. 医療応用

11.4 ハイパーパラメータの選択

- ハイパーパラメータの影響
  1. アルゴリズム実行時間、メモリコスト
  2. モデルの性能

- ハイパーパラメータ選択の手動 or 自動
  - 手動: ハイパーパラメータとモデルに対する理解が必要
  - 自動: 上記を理解する必要はないが、計算コスト大

- ハイパーパラメータの例
  - 隠れユニットの数、学習率、畳み込みカーネルの幅、暗黙的なゼロパディング、重み減衰の係数、ドロップアウトの割合

11.4 ハイパーパラメータの選択

- 手動でのハイパーパラメータ調節
  - 制約: 実行時間とメモリ
  - 目的: モデルの有効容量を、タスクの複雑度に適合するように調整して、汎化誤差を最小化すること
  - モデルの有効容量を左右する三要素
    1. コスト関数を最小化する学習アルゴリズムの能力
    2. 正則化
    3. モデルの表現能力（層・ユニットの数）
       - 深層学習には当てはまらない?
         (https://www.slideshare.net/DeepLearningJP2016/dla-bayesian-perspective-on-generalization-and-stochasticgradient-descent-81735888)

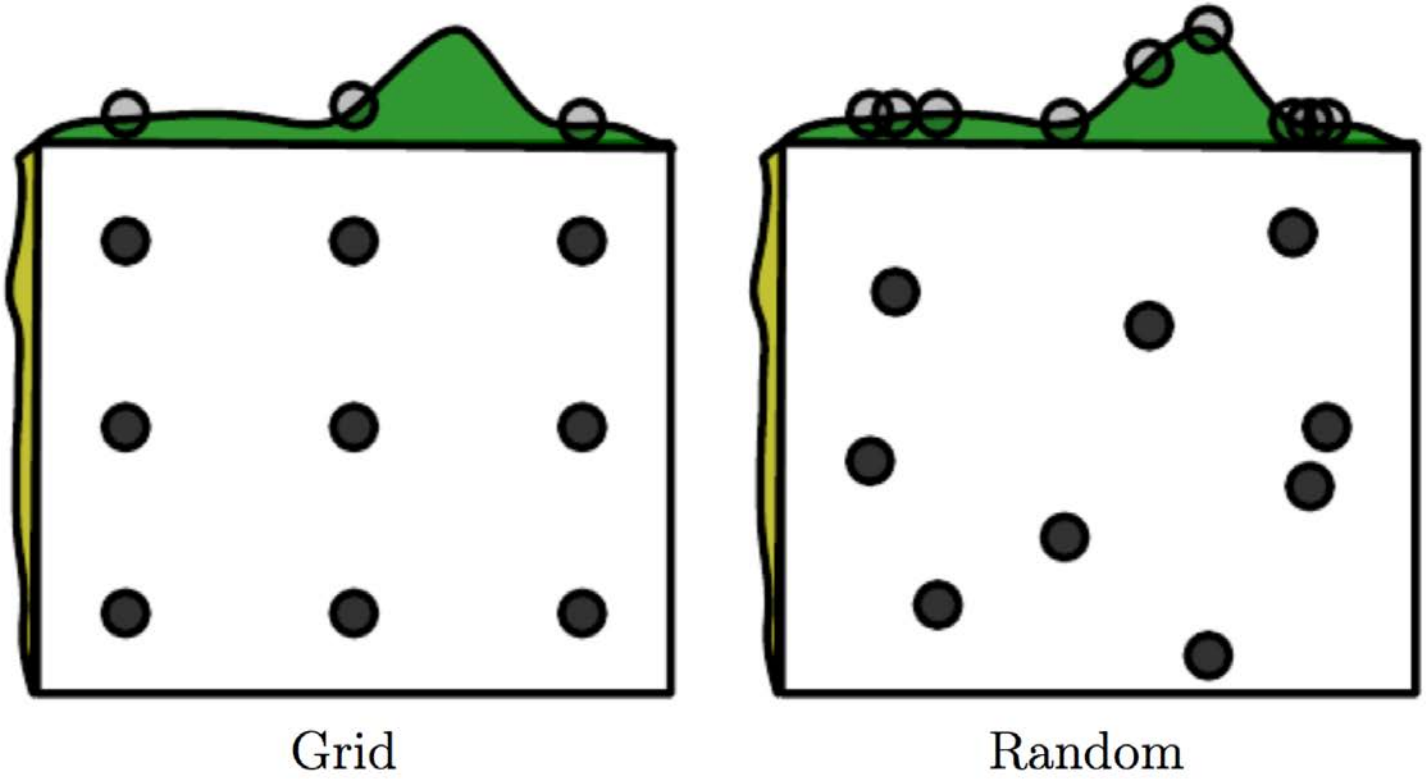
11.4 ハイパーパラメータの選択

- 自動でのハイパーパラメータ調節を行うアルゴリズム
  - ロジスティック回帰やSVMと比べて深層学習のハイパラは多いので、探索が大変
  - そこで、ハイパーパラメータ最適化アルゴリズムを用いると便利

- グリッッドサーチ
  - 手順
    1. 各ハイパーパラメータについて、探索対象となる値の集合を選択
    2. 各ハイパーパラメータの集合のデカルト積のすべての要素を探索
  - 欠点: 計算コストが指数関数的に増加する
    - m個のハイパーパラメータがあり、それぞれが最大でn個の値を取る場合、必要となる学習と評価の試行回数は$O(m^n)$で増加する

11.4 ハイパーパラメータの選択

- ランダムサーチ
  - 各ハイパーパラメータが従う周辺分布を定義して、そこからサンプリングする
  - 影響力のないハイパーパラメータに対して無駄な探索を行わないので、グリッドサーチよりもずっと早く収束する



11.4 ハイパーパラメータの選択

- モデルに基づくハイパーパラメータの最適化
  - 決定変数をハイパーパラメータ、検証集合誤差をコストとした、最適化問題を考える
  - ナイーブに実行するのは勾配が求まらず難しい
  - そこで、検証集合誤差のモデルを作成し、そのモデル内で最適化を実行する手法がある
    - ベイズ回帰モデルにより、検証集合誤差の期待値とその不確実性を推定
    - その結果、探索と活用のトレードオフを伴う最適化問題に帰着
      - Spearmint, TPE, SMAC
  - 重要な研究分野ではあるが、まだ成熟しきっておらず、深層学習への適用を明確に推奨することはできない
  - 参考: https://www.slideshare.net/hoxo_m/ss-77421091

- 手動探索と自動探索の比較
  - 手動探索では、学習の途中で一部のハイパーパラメータが全く役に立たないことを知って、事件を停止することができるというメリットがある
  - しかし、自動探索でも同様の機能（有望ではない実行中の実験を「停止」する）を持つアルゴリズムが提案されている

11.5 デバッグの戦略

- 機械学習システムがうまく動作しない時、アルゴリズム自体の問題なのか、アルゴリズムの実装にバグがあるのかを判断するのは難しい
  - アルゴリズムの動作が意図したものかを知るのが難しい
    - 前例がないタスクで、5%のテスト誤差を達成したとして、これが最適な性能の動作かはわからない
  - 機械学習モデルの一部が壊れているとしても、他の部分が適応することで、許容できる性能を発揮してしまう場合がある

- 二つの基本的なデバッグ戦略
  - 正しい挙動が実際に予測できるような簡単なタスクを設計する
  - ニューラルネットワークの一部に対して独立に実行するテストを設ける

11.5 デバッグの戦略

- 重要なデバッグテスト例
  - 実行中のモデルの可視化
    - 物体検出の場合はフィルターの可視化
    - 生成モデルの場合は生成されたサンプルの観察
  - 最悪の失敗の可視化
    - 例えば、OCRでもっとも確信度の高い失敗をしている画像を見てみると、croppingの失敗によって文字の一部が除外されていることが判明した（後述）
  - 訓練誤差とテスト誤差を用いたソフトウェアに関する推論
    - 基本となるソフトウェア自体が正しく実装されているか知るのは難しい
    - 訓練誤差 < テスト誤差なら、訓練手順が正しく動作している可能性が高い
  - 小さなデータ集合への適合
    - 訓練集合において誤差が大きい場合、過少適合なのかソフトウェアの不具合なのか知りたい
    - たとえば事例が一つだけの分類タスクを解けないとしたら、ソフトウェアの不具合が起きている可能性が高い

11.5 デバッグの戦略

- 重要なデバッグテスト例
  - 逆伝搬の微分と数値微分の比較
    - 勾配計算を自分で実装すると、勾配の式を間違えることがある
    - 自動微分の実装で計算された微分と、有限差分法で計算された微分を比較する
    - 有限差分法: $f'(x) \approx \frac{f(x+\varepsilon)-f(x)}{\varepsilon}$
  - 活性化と勾配のヒストグラム監視
    - 隠れユニットがどれくらい飽和しているか、常にオフになっていないか
    - 勾配はパラメータの大きさの1%程度になってほしい(not 50%や0.001%)
      - ただし、データがNLPのようにスパースな時、めったに更新されないパラメータがあることに注意
  - アルゴリズムの特性による保証
    - アルゴリズムの1ステップ後に目的関数が決して増加しないこと
    - 収束の時にすべての変数に関する勾配がゼロになること

11.6 例: 複数桁の数字認識

- 実装の例として、Street view address number transcription systemをとりあげる
  - Googleマップに建造物の[GPS座標、住所]を追加するアプリ
    - ストリートビューカーは建物の写真を撮り、各写真に対応するGPS座標を記録
    - CNNが各写真から住所を認識し、Googleマップデータベースに記録

11.6 例: 複数桁の数字認識

- プロジェクトの工程
  1. 評価指標の選択
     - 地図なので、高い精度を保ちたい
     - 正解率を98%にした状態で網羅率を最適化する
  2. ベースラインシステムの確立
     1. 出力はaddressなので系列だったが、当時CNNに系列を出力させるのは一般的でなかった
     2. しかしCVタスクなので、ReLUを持つCNNから始めた
     3. ベースラインの改善により、理論上問題のないアプローチができた
  3. 訓練誤差とテスト誤差の比較
     - 数千万件のラベル付きデータを使ったので、ほぼ差がなかった
  4. デバッグ
     1. この時点で網羅率は90%以下だったので、過少適合かデータの質に問題がある
     2. 最悪の結果の可視化を行なったところ、画像のクロッピングによって住所が書かれた文字列の一部が消えていることがわかった。
     3. 修正したところ、網羅率が10%あがった
  5. ハイパラ調整
     1. 訓練誤差とテスト誤差がほぼ同じだったので、モデルの容量を増やした

参考文献 1

- [Soudry+ 2017] The Implicit Bias of Gradient Descent on Separable Data
- [Kingma+ 2014] Semi-supervised learning with deep generative models
- A Bayesian Perspective on Generalization and Stochastic Gradient Descentのスライド
  - https://www.slideshare.net/DeepLearningJP2016/dla-bayesian-perspective-on-generalization-and-stochasticgradient-descent-81735888
- 機械学習のためのベイズ最適化入門
  - https://www.slideshare.net/hoxo_m/ss-77421091

参考文献 2

- Deep Learning
  - Ian Goodfellow, Yoshua Bengio, Aaron Courville
  - 日本語版
    https://www.amazon.co.jp/%E6%B7%B1%E5%B1%A4%E5%AD%A6%E7%BF%92-Ian-Goodfellow/dp/4048930621