PythonとKerasによるディープラーニング
4.5 機械学習の一般的なワークフロー

17T4028N 菊田尚樹

機械学習のワークフロー

①問題の定義

②指標の選択

③評価方法の決定

④データの準備

⑤小さなモデルの作成

⑥モデルのスケールアップ

⑦正則化とハイパーパラメータのチューニング

①問題の定義

⑴ 入力データは何か、予測しようとしているものは何か

⑵ 直面している問題はどのような種類のものか　分類？回帰？生成？…

→モデルのアーキテクチャや損失関数を選択する目安

□この段階での仮説

・出力は入力から予想できる

・利用可能なデータは入力と出力の関係を学習するのに十分揃っている

↓　※実際に動くモデルが完成するまでわからない

<u>非定常問題に注意</u>

求められる内容が絶えず変化し、時期に左右される問題を扱う際は学習させるデータに注意しなければならない

(例)ファッション関連　夏のデータで冬のトレンド予測はできない…

適切なデータを適切な方法で学習させる必要がある

②指標の選択

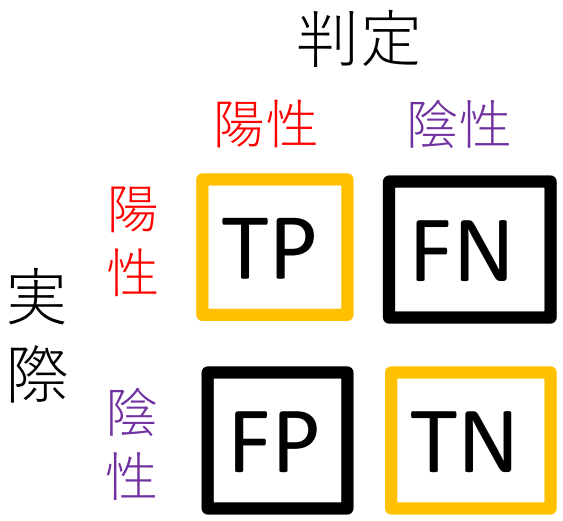
何を観測するのか、何をもって成功とするのか

分類問題の場合

$$\text{適合率} = \frac{TP}{FP+TP}$$

$$\text{再現率} = \frac{TP}{FN+TP}$$

$$\text{正解率} = \frac{TP+TN}{TP+FP+FN+TN}$$



適合率と再現率はトレードオフの関係　適合率を上げようとすれば再現率は下がる

逆も然り

$$\text{F値} = \frac{\text{適合率} * \text{再現率}}{\frac{1}{2} * (\text{適合率} + \text{再現率})}$$

←調和平均

③評価方法の決定

現在の進捗状況を評価する方法を決定する
以下の三つのうちから選択

・ホールドアウト法の検証データセットの確保
　　データが十分にある場合

・K分割交差検証を実行
　　ホールドアウト法を確実に行うにはサンプルが少ない場合

・反復的なK分割交差検証を実行
　　利用可能なデータ数が少ない場合にモデルの評価を正確に行う

④データの準備

データのフォーマットを機械学習モデルに供給できるものにする必要がある

～ディープニューラルネットワークの場合～

・データはテンソルの形

・テンソルの値は０～１や－１～１のように範囲を統一した状態

・特徴量によって値の取りうる範囲が異なる場合は正規化して上記に持ち込む

・利用可能なデータが少ない場合は特徴エンジニアリングが必要な場合もある

⑤小さなモデルの作成

目標：ベースラインを超える程度の性能をもつモデルの作成

　ベースラインを超える→統計的検出力を実現する

・最後の層での活性化

　　分類ではシグモイドなど　回帰では使わない

・損失関数の選択

　　扱う問題に適したもの

・最適化の設定

　　オプティマイザは何か、学習率は何か

何度も試してうまくいかない場合は当初の仮説が成り立っていない場合がある

　仮説

　・出力は入力から予想できる

　・利用可能なデータは入力と出力の関係を学習するのに十分揃っている

成り立っていないのであれば設計段階に戻る

⑥モデルのスケールアップ

ベースラインを超えるモデルが十分な性能を持っているかが課題

理想的なモデルは学習不足と過学習の境界線上に存在

　境界線を見つけるためにはまず過学習させ線をまたいでみることが必要

過学習のさせ方

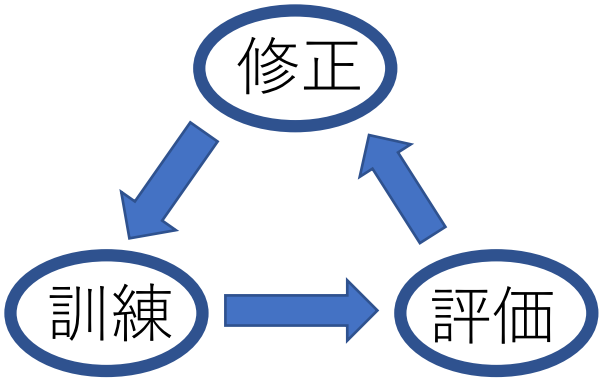
・層を追加する

・層を大きくする

・訓練のエポック数を増やす

検証データで試し、性能が低下し始めたら過学習している証拠

⑦正則化とハイパーパラメータのチューニング



最も時間をかけるステップ

モデルが性能を出し切れるまで繰り返す

試してみるべきもの

- ドロップアウトの追加
- 別のアーキテクチャ
- L1/L2正則化の追加
- 別のハイパーパラメータで試す(学習率、ユニット数)
- 新たに特徴量を追加、不要な特徴量の削除

⑦正則化とハイパーパラメータのチューニング

注意すべきこと

検証プロセスでのフィードバックを用いてチューニングするたびに検証プロセスの情報がモデルに漏れ出す

→検証データに対し過学習してしまい信頼性がなくなる

満足のいく設定ができたら

利用可能なすべてのデータ(訓練データ、検証データ)でモデルを訓練し、テストデータを使って評価する

テストデータでの評価結果が検証データでの評価を著しく下回るならば検証データに対して過学習してしまっている可能性大