Python ではじめる機械学習

2.3 教師あり機械学習アルゴリズム

15T4032G 芝山直希

2018 年 4 月 20 日

2.3.1 サンプルデータセット
低次元データセット

1 データが持つ特徴量の数が少ないもの
簡単に可視化できる

- forge
  - 2 つの特徴量を持つ 26 のデータが 2 クラスに分けられている合成データセット
- wave
  - 1 つの特徴量と連続値のターゲット変数を持つ合成データセット
  - 回帰アルゴリズムの紹介時に用いる
- 低次元データセットに対する直感が、高次元データセットで通用するとは限らない

高次元データセット

1 データが持つ特徴量の数が多いもの
実問題から取って来る

- ウィスコンシン乳癌データセット (cancer)
  - 30 の特徴量を持つ 569 データからなるデータセット
  - 乳癌の腫瘍を計測したもの 良性か悪性かでラベル付けがされている
  - 測定結果を与えると、悪性腫瘍かどうかを予測するよう学習するのが目標
- boston_housing
  - 13 の特徴量を持つ 506 のデータセット
  - 1970 年代のボストン近郊の住宅地の住宅価格の中央値を予測するのが目標
  - ここでは、各特徴量間の積 [1] を元に特徴量を 104 種類に拡張したデータセットも用いる

[1] 交互作用という。

2.3.2 k-最近傍法

k-NN(k-最近傍法) アルゴリズム。最も単純な学習アルゴリズム。

- 訓練データセットそのものをモデルとして格納する
- 予測法…入力されたデータとの距離が最も近い k 個の近傍点による多数決
- 実用上 k は 3〜5 程度でも十分 要調整
- 分類が主なタスクだが、回帰を行うものもある
  - 回帰は k 個の最近傍点の平均値を予測値とする
- 利点…理解しやすい単純なモデル、あまり調整しなくても良い結果が出る、モデル構築が速い
- 欠点…大きな訓練データセットを使うと予測が遅い、特徴量数が数百以上だと機能しない、疎なデータセットに弱い

処理速度が遅く大規模なデータセットに対して使えないため、ほとんど使われていない

2.3.3 線形モデル

ターゲットを特徴量の線形和で表現できるという仮定のもとに成り立つモデル

- p+1 個の特徴量を用いて、回帰の場合次の予測式に基づいて予測する

$$\dot{y} = w[0] \times x[0] + w[1] \times x[1] + \cdots + w[p] \times x[p] + b \qquad (1)$$

  - ただし、x は入力データ、w、b を訓練データセットから学習する
  - 分類の場合、条件式 $(1) > 0$ による 2 クラス分類を行う
- 回帰、分類のどちらを行うかはアルゴリズム依存
- 利点…訓練・予測が速い、比較的理解しやすい予測手法、高次元データセット・疎なデータセットにも適用可能
- 欠点…係数の値の根拠があまり明確でない、低次元データセットでは他のモデルの方が汎化出来ていることがある

線形モデルの回帰アルゴリズム

- 線形回帰 (通常最小二乗法)
  - 訓練時、予測と回帰ターゲット y との平均二乗誤差 [2] が最小になるように w, b を学ぶ
  - パラメータを持たないため、モデルの複雑さを制御できない
- リッジ回帰
  - 訓練時、予測とターゲットとの平均二乗誤差および w を可能な限り小さくするように w, b を学ぶ
  - パラメータ alpha を持ち、正則化 [3] の度合いを制御できる。0 に近づくほど w が線形回帰での学習結果に近づく (デフォルトは 1)

[2] 誤差を二乗した値の平均値。

[3] 過剰適合しないように、明示的にモデルを制約すること。リッジ回帰では L2 正則化という方式を用いている。

- Lasso
  - リッジ回帰と同様に w, b を学習するが、w に対する制約のかけ方が異なる (L1 正則化)
  - L1 正則化により、一部の特徴量にかかる係数が 0 になり、重要な特徴量が明らかになる
  - パラメータ alpha を持ち、小さくするほどより複雑なモデルに適合できる。0 に近づくほど線形回帰に近づく

線形モデルの回帰アルゴリズムの中では、リッジ回帰がおすすめだが、特徴量が多く、重要な特徴量は少ないと思われる時、及び解釈しやすいモデルがほしい時に Lasso を利用すると良い。
実用上はリッジ回帰と Lasso の組み合わせが良いが、パラメータ調整コストがかかる

線形モデルでのクラス分類

2 クラスの決定境界となる線形関数を訓練データから学習する

- ロジスティック回帰、及び線形サポートベクタマシン (linear SVM) が最も一般的
- これらはパラメータ C を持ち、正則化の度合いを調整できる また、penalty パラメータで正規化の方式を指定できる
- 多くの線形クラス分類アルゴリズムはモデル 1 つにつき 2 クラスまでしか分類できない
  - 多クラス分類をしたい場合、1 対その他アプローチという、「あるクラスであるかどうかを判別する」分類器を複数利用し、最も高い値を得られたクラスとする方式が一般的
  - なお、ロジスティック回帰は多クラス分類に対応している

2.3.4 ナイーブベイズクラス分類器

線形モデルによく似たクラス分類器

- クラスに対する統計値を特徴量ごとに収集し、パラメータを学習する
- scikit-learn では GaussianNB、BernoulliNB、MultinomialNB が実装されている
  - GaussianNB は任意の連続値データを受け付ける。高次元データに対して用いられることが多い
  - BernoulliNB は 2 値データを想定している。疎なカウントデータに対して用いられることが多い
  - MultinomialNB はカウントデータ [4] を想定している。BernoulliNB と同様の用途だが、一般的にはこちらのほうが若干高性能であり、比較的多数の非ゼロ特徴量を持つデータには特に有効

[4] 個々の特徴量が何らかのカウンタであるデータ。

- GaussianNB 以外の 2 つはパラメータ alpha でモデルの複雑さを制御できる 影響力は低め
- 線形モデルと共通する利点・欠点を持つ
- 線形モデルに比べ訓練がより速いが、汎化性能でわずかに劣ることが多い

線形モデルでも時間がかかりすぎるレベルの大規模なデータセットに対するベースラインとして有効

2.3.5 決定木

クラス分類と回帰に広く用いられているアルゴリズム

- 正解に最速で到達できるような一連の Yes/No 型質問（テスト）からなる木構造を学習する
- 葉がデータセット内の 1 つのデータに対応するまで学習を続けると過剰適合するため、枝刈りが必須
  - 事前枝刈り…構築段階で早めに木の生成を止める
  - 事後枝刈り（枝刈り）…木構造を作ってから情報の少ないノードを消す scikit-learn では実装されていない
- 3 つのパラメータ max_depth(木の深さの上限)、max_leaf_nodes(葉の数の上限)、min_samples_leaf(葉に含まれる訓練データポイント数の下限) で事前枝刈りのタイミングを制御する
- 利点…データのスケールに依存しないモデル、可視化が容易で (浅いモデルなら) 専門家でなくてもわかる
- 欠点…性質上訓練データにない領域での回帰 (外挿) が出来ない、枝刈りしても過剰適合しやすく、汎化性能が低い

2.3.6 決定木のアンサンブル法

複数のモデルを組み合わせて、より強力なモデルを構築する
様々なデータセットに対するクラス分類・回帰に有効であることが明らかなもの：

1. ランダムフォレスト
   - 「一部のデータに過剰適合しているが、他のデータは上手く予測できる」、過剰適合している方向が異なる決定木を複数用意し、平均を取る
2. 勾配ブースティング回帰木
   - 一つ前の決定木の誤りを修正するような浅めの決定木を順番に作り、組み合わせる
   - クラス分類にも利用可能

ランダムフォレストの特徴

学習法：

1. 訓練データセット内から、重複ありでランダムにデータを複数選択し、訓練データセットと同じ長さだが偏りのあるデータセットを作る (ブートストラップサンプリング)
2. 使用する特徴量サブセットをランダムに選択し、その特徴量を利用する最適な決定木を 1 で作ったデータセットを用いて構築
3. 上記工程を作成する決定木の本数分行う

主要パラメータ：n_estimators(木の本数)、max_features(1 本の木が利用する特徴量数)、事前枝刈りパラメータ

- max_features は個々の木の乱数性に影響し、訓練データセットへの適合度合いを左右する
- n_estimators は性能に直結 できる限り大きくする
- n_jobs パラメータを利用すれば、簡単に学習の並列化ができる

利点：高い性能、さほどパラメータチューニングしなくても使える、並列化が簡単、決定木の利点の多くを持つ
欠点：メモリを多く消費する、訓練・予測に時間がかかる、テキスト等の超高次元かつ疎なデータではうまく行かない傾向あり

勾配ブースティング回帰木の特徴

- (デフォルトの状態では) 乱数性を持たないが、早い段階で強く事前枝刈りする
- 浅い決定木のような簡単なモデルを多数組み合わせ、高い性能を出す

主要パラメータ：n_estimators(本数、増えるほど複雑化)、learning_rate(過去の決定木の誤りをどの程度補正するか)、事前枝刈りパラメータ

- n_estimators と learning_rate 間には強い相関がある
- 事前枝刈りパラメータは max_depth を 5 以下にするのが一般的

利点：最強クラスの性能、ランダムフォレストよりメモリ消費量が少ない、予測が高速、決定木をベースとするモデルの利点を持つ
欠点：訓練に時間がかかる、パラメータ設定に細心の注意を払う必要がある、決定木ベースのモデルの欠点を持つ

2.3.7 カーネル法を用いたサポートベクタマシン

SVM。線形 SVM を拡張したもの ここではクラス分類器 SVC について述べるが、回帰 (SVR) に対しても適用可能

- 適切な非線形特徴量（特徴量間の交互作用など）を追加することで線形モデルはより強力になる
- データを拡張せずに、拡張された特徴表現上でのデータ間の距離を計算する技法カーネルトリックが用いられている

主要カーネル (高次元空間へのマッピング方式) :

- 多項式カーネル (polynomial kernel)…元の特徴量の特定次数までの全多項式を計算
- ガウシアンカーネル (放射基底関数カーネル、Radial Basis Function)…無限次元に対応、直感的にいうとテイラー展開のような計算をしている

SVC の仕組みと主なパラメータ

- 訓練データ内の個々のデータが決定境界の表現にどの程度重要か学習する
- 多くの場合、2 クラスの境界に位置する一部のデータ (サポートベクタ) が重要視される
- 予測時は各サポートベクタとの距離と、学習した各サポートベクタの重要度を用いて分類する

主要パラメータ：

- カーネルの種類とカーネル固有パラメータ
  - RBF カーネルの場合パラメータ gamma(どの程度の距離を「決定境界に近い」と認識するか) を持つ
- 正則化パラメータ C
  - カーネル固有パラメータの中に、モデルの複雑さに作用するものがある場合、それと同時に調整する

SVM の利点、欠点

利点：

- 様々なデータセットに対して上手くいく強力さ
- 次元の大きさに関わらず上手く機能する

欠点：

- データのスケールに敏感であり、前処理とパラメータ調整を注意深く行う必要がある
- サンプルの個数が超巨大 (100000 程度) になると上手く機能しない (実行困難)
- 検証及び予測根拠の理解が困難であり、非専門家への説明に苦労する

特徴量が同じスケールになる場合に有効

2.3.7 ニューラルネットワーク (ディープラーニング)

比較的簡単な多層パーセプトロン (multilayer perceptron: MLP、(フィードフォワード・) ニューラルネットワークともいう) によるクラス分類・回帰について触れていく
概要：線形回帰同様の重み付き和の計算を繰り返し行い、最終結果を出力する

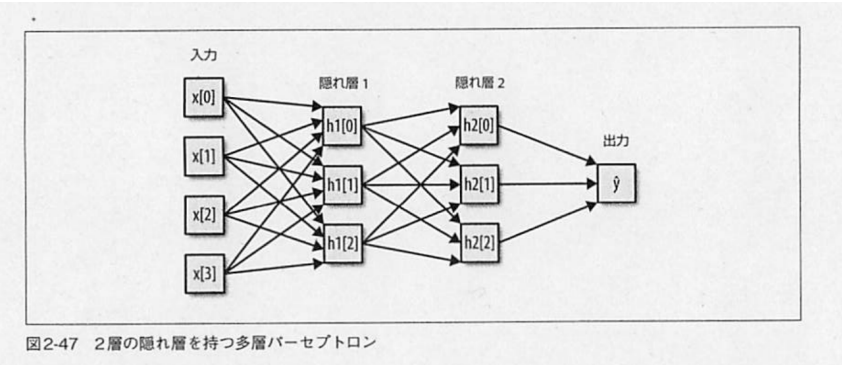

図2-47　2層の隠れ層を持つ多層パーセプトロン

- 入力→隠れ層 1 の全ユニット→隠れ層 2 →…→隠れ層 n →出力 という流れで重み付き和が計算されていく
- 学習すべき係数は学習前に乱数で初期化され、数が非常に多い (左図の矢印の数にほぼ等しい)
- 隠れ層のユニット (隠れユニット) の値の計算時に、非線形関数 (活性化関数) を計算結果に適応し複雑な関数を学習させる

ニューラルネットワークのパラメータ

ニューラルネットワークのパラメータ調整は「1 つの技芸」

- 隠れ層の数、隠れ層 1 層あたりのユニット数…増やすほど複雑化
- 正則化パラメータ alpha
- 活性化関数…relu($y = max(x, 0)$)、tanh($y = tanh(x)$) が一般的
- 学習アルゴリズム
  - adam…ほとんどのケースでうまくいくが、データのスケールに敏感 デフォルト
  - libfgs…頑健だが、モデルやデータセットが大きくなると時間がかかる
  - sgd…より高度 設定可能なパラメータが増え、調整の手間が増える

過剰適合できる大きさから始め、汎化性能を強化する方向にパラメータを変更するのが一般的

ニューラルネットワークの利点、欠点

利点：

- 他のアルゴリズムに勝る性能
  - 十分な時間・データ量、及び適切に調整されたパラメータがあれば、他の機械学習アルゴリズムに勝てることが多い
- SVM 同様、特徴量が同じスケールであるデータに強い

欠点：

- 大きく強力なモデルほど、訓練に時間がかかる
- データのスケールが揃ってなければ性能の全てを発揮できず、慎重な前処理が不可欠

他のディープラーニングライブラリの方が scikit-learn より柔軟かつ大きなモデルを使用できる

- keras, lasagna, tensor-flow が Python ユーザ間で広く使われている
- 人気ディープラーニングライブラリは GPU での計算に対応