レポート_機械学習

線形回帰モデル

要点

回帰
ある入力から **連続した** 出力値を予測する。直線で予測するのが線形回帰で、直線以外で予測するのが非線形回帰と呼ぶ。

線形回帰モデルは教師あり学習であり、教師データと呼ばれる (説明変数, 目的変数) : (x_i, y_i) の組み合わせから学習モデルを作成し、未知のデータについての予測を行う。予測値 $\hat{y}$ は、以下の式で表される。

$\hat{y} = \Sigma_{j=1}^{m} w_j x_j + w_0$

- w : パラメータ
- x : 説明変数（特徴量）
- 慣例として、予測値にはハットをつける。

説明変数が1つの場合、 **単回帰モデル** と呼ぶ。
説明変数が複数（多次元）の場合、 **重回帰モデル** と呼ぶ。

教師データを学習用と検証用に分け、学習用データでモデルを作成し、検証用データでモデルの性能を評価する。データを分割する理由としては、結果がわかっている検証用のデータを入力とすることでモデルの **汎化性能** を測定するため。機械学習の目的は未知のデータについての予測を行うもので、既知のデータで精度が高く出ても意味がない。

データとモデル出力の二乗誤差の和を **平均二乗誤差** （MSE: Mean Squared Error）と呼び、特に平均二乗誤差を最小とするパラメータを探索する手法を **最小二乗法** と呼ぶ。平均二乗誤差は、以下の式で表される。

$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)^2$

線形回帰モデル等の機械学習によく使われるPythonのライブラリとして Numpy, Pandas, Scikit-learn がある。

- Numpyは多次元配列の扱いを得意とするもので、行列演算や数学に関する多数の関数が用意されている。
- PandasはExcelのような二次元データの扱いを得意とし、二次元データの抽出・結合・拡張・削除・集計といった操作を行うことができる。
- Scikit-learnは機械学習関連アルゴリズムが複数用意されている。学習データを訓練用・テスト用に分割したり、回帰・分類に関する様々なアルゴリズムや、前処理関連や評価用の関数などが用意されている。

実装演習

- NumPyによる演習
- Scikit-learnによる演習：ボストンの住宅データセット

追加レポート

- 共分散について

非線形回帰モデル

要点

データ構造を線形で捉えられる場合は、限定される。

未学習（underfitiing）と過学習（overfitting）

- 未学習が発生するのは、学習データの誤差が十分小さくないとき。
  - 対策として、表現力の高い（つまり、より複雑な）モデルを利用する。
- 過学習が発生するのは、学習データに対して十分小さな誤差が得られたが、テストデータの誤差が大きくなったとき。
  - 対策としては、以下が考えられる。
    1. 学習データの数を増やす。
    2. 不要な説明変数を削除する。
    3. 正則化(Ridge, Lasso等)を行う。
- 学習データの誤差（バイアス）と、過学習による誤差（バリアンス）はトレードオフの関係にあるため、上記の対策を行いながら最適解を見つけていく。

正則化項には、L2ノルムとL1ノルムがある。

- L2ノルムを利用するのが **Ridge推定量** で、パラメータを0に近づけるようにする。
- L1ノルムを利用するのが **Lasso推定量** で、いくつかのパラメータを完全に0にしてしまう（スパース推定）。

- 学習データだけでなく未知のデータへの予測性能を **汎化性能** と呼び、この汎化性能が高いモデルが良いモデルといえる。汎化誤差は通常、検証用データでの性能を測ることで推定する。
- よく用いられるのが **交差検証(cross validation)** で、データ全体を学習用と検証用に分割する。特に、データ全体をk分割しそのうちの1データセットを検証用として交差検証を行い、次に別のデータセットで交差検証を行い、これをk回繰り返すことを **k分割交差検証** と呼ぶ。

モデルのパラメータを複数指定し、その全ての組み合わせについて学習を行い評価する方法を **グリッドサーチ** と呼ぶ。

実装演習

- NumPyによる演習

- scikit-learnによる演習

ロジスティック回帰モデル

要点

ロジスティック回帰モデルとは、0か1の2値分類に用いられる。
入力 x 、パラメータ w としたとき、出力 y は以下の線形結合により求められる。

$\hat{y} = w^T x + w_0 = \Sigma_{j=1}^{m} w_j x_j + w_0$

出力は $y = 1$ となる確率で、 **シグモイド関数** が用いられる。シグモイド関数は、以下の式で与えられる。

$sigmoid(x) = \frac{1}{1+exp(-ax)}$

シグモイド関数による分類では、出力の範囲は0以上1未満の確率で、0.5以上だと $y = 1$ 、0.5未満だと $y = 0$ に分類される。

最尤推定とは、実測値からその値を生成した尤もらしい分布を推定すること。

勾配降下法とは、反復学習によりパラメータを更新していく方法。 **学習率** を調整して、勾配が0になるまで更新を続けていく。

- 学習率が高すぎる：学習が収束しなくなる
- 学習率が低すぎる；学習の進みが遅くなる

以上から、現実的な更新回数で学習が収束するような適切な学習率を設定することが重要である。

勾配降下法では、パラメータ更新にN個すべてのデータの和を求める必要がある。Nが巨大になると、計算に膨大な時間がかかる。そこで、 **確率的勾配降下法** (SGD) を用いる。これは、毎回のパラメータ更新の際にデータを1つランダムに選択することで、計算時間を抑えることが可能。

分類の評価は、 **適合率** (Precision), **再現率** (Recall) があり、これらはトレードオフの関係となっている。両者の調和平均として、 **F値** という指標があり、これも評価方法としてよく用いられる。

- 適合率：陽性に分類された（真陽性＋偽陽性）中で、本当に陽性だった（真陽性）割合
- 再現率：本来は陽性だったもの（真陽性＋偽陰性）の中で、そのうち本当に陽性だった（真陽性）割合

実装演習

- NumPyによる演習
- タイタニックの乗客データ

追加レポート

- 等高線の描き方について

主成分分析

要点

- 主成分分析とは、主に教師なし学習で多変量変数の数を減らし、かつ情報の損失をなるべく抑えたいときに利用される手法。変数を減らすことでデータ分析を行いやすくしたり、可視化させることができる。
- 主成分分析では、情報の量を **分散の大きさ** と捉える。つまり、分散が大きいほどそのデータが重要であるとしている。具体的には、固有値と固有ベクトルを求め、固有値を昇順で並べたときの固有ベクトルを順に **第k主成分** と呼ぶ。特に、最も情報量の多い主成分を第1主成分と呼ぶ。
- **寄与率** とは、第k主成分の分散の、全分散に対する割合で、第1主成分が最も大きく、第2、第3、...になるにつれて小さくなる。すべての主成分の寄与率の和は1となる。また、第k主成分までの寄与率の合計値を **累積寄与率** と呼び、累積寄与率を指標として、どこまでの主成分を採用するかを判断することもある。

実装演習

- NumPyによる演習
- scikit-learnによる演習

アルゴリズム

要点

- k-近傍法（KNN）
  教師あり学習の分類アルゴリズム。あるデータ点の近い順からk個のデータを見つけ、その見つかったデータがどのラベルに属するかを多数決で決定する。例えば、k=5である点から近い順を見たときラベル0のデータが3個、ラベル1のデータが2個だった場合、その点はラベル0に属する。
- k-means
  教師なし学習のアルゴリズム。以下の手順によって、各データをクラスタに割り当てる。
  1. 各クラスタ中心の初期値をランダムに設定する。
  2. 各データ点に対して、1で設定したクラスタ中心との距離を計算し、最も距離が近いクラスタに割り当てる。
  3. 2ですべてのデータ点がクラスタに割り当てられたので、各クラスタについて中心点を求める。
  4. 3で求めた中心点と前回設定した中心点と同じなら終了し、中心点が決定される。そうでなければ、2に戻り新たなクラスタ中心でデータ点の再割り当てを行っていく。

実装演習

- NumPyによる演習：k-means
- scikit-learnによる演習：k-means
- NumPyによる演習：k近傍法

サポートベクターマシーン

要点

- サポートベクターマシーン（SVM）は、教師あり学習のクラス分類を行う際に利用されるアルゴリズムである。線形モデルによる2値分類をする際に、決定境界をどう決めていくかを考える。

SVMでは、マージン最大の手法で決定境界を決めていく。

- 線形判別関数を $w^T x + b = 0$ としたとき、線形判別関数と最も近い点との距離を **マージン** と呼ぶ。このマージンが最大となる線形判別関数を求めていく。
- SVMの目的関数と制約条件は以下となる。
  - $min_{w,b} \frac{1}{2}||w||^2$
  - $t_i(w^T x_i + b) \geq 1 \ (i = 1, 2, ..., n)$
- マージン上にあるデータのみが予測に影響を与える。これを **サポートベクタ** と呼び、逆にサポートベクタ以外のデータ点は不要になる。

- サンプルデータを線形分離できないとき、誤差を許容し、誤差に対してペナルティを与える。これを **ソフトマージンSVM** と呼ぶ。
- ソフトマージンSVMの目的関数と制約条件は以下となる。
  - $\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$
  - $t_i(w^T x + b) \geq 1 - \xi_i$
  - ここで C は、トレードオフを制御するパラメータである。このパラメータの大小で決定境界が変化する。 C が小さいときは、誤差をより許容し、大きいときは誤差を許容しなくなる。

- 線形分離ができないとき、特徴空間に写像し、その空間で線形に分離することができる。このとき行われる手法を **カーネルトリック** と呼び、高次元ベクトルの内積をスカラー関数で表現し、計算コストを抑えることができる。
- 非線形カーネルを用いた分離を行い、その際によく使われるのが **放射基底関数カーネル (Radial Basis Function : RBF)** （又は **ガウシアンカーネル** とも呼ぶ）である。ガウシアンカーネルは、以下の式で与えられる。
  - $k(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$

実装演習

- NumPyによる演習