機械学習　レポート

2024 年 06 月 09 日 新規作成

1. パターン認識

1.1. k 近傍法（kNN）

　k 近傍法とは分類に利用される教師あり学習のアルゴリズム。判別したいデータから最も距離が近いデータ点を k 個取り、その中の多数決で分類する。

手順

1. 学習データを数値化する。（カテゴリ変数など）
2. 分類したいデータと学習データの距離をユークリッド距離などで計算する。
3. 分類したいデータから距離が近い順に学習データを k 個取りだしその多数決で分類する。
4. 最適な k を調べる。学習データとテストデータに分け性能評価する。

一般に k が大きいとデータのノイズに対し頑健になるが、その分のデータが計算に入るのでクラス間の違いが明確に分からなくなる。k が増えると計算量も増える。

1.2. 近似最近傍探索

近似最近傍探索は効率的に近似的な最近傍を検索するための技術。高次元データや大規模データセットに対して精度を犠牲にする代わりに検索時間を大幅に短縮できる。高次元データの次元の呪い、大規模データセットの膨大な計算時間を回避する。

手法

- 局所感度ハッシュ（Locality-Sensitive Hashing、LSH）
  特定の距離尺度に対して類似したデータポイントを同じバケットにハッシュする。
  クエリポイントと同じバケット内のポイントを検索することで計算量を削減する。
- k-d Tree
  データポイントを分割しながら空間を二分していく。低次元データには効果的だが高次元データには適していない。
- グラフベースの手法
  データポイントをノードとして扱い、エッジで接続されたグラフを構築する。クエリポイントから始めて近いノードへと徐々に移動し近似最近傍を見つける。

評価指標

- 精度
- 検索時間
- メモリの消費量

応用例

- 類似画像の検索
- レコメンデーション
- 単語埋め込みベクトルの類似性検索

1.3. 距離計算手法

1. コサイン距離
   2 つのベクトルの角度の違いを測定する方法。ベクトルの大きさは無視される。

$$D_{cos}(a,b) = 1 - \frac{a \cdot b}{\| a \| \| b \|}$$

   ここで、$\| a \|$と$\| b \|$はベクトルのノルム。
2. ユークリッド距離
   2 つの点間の直線距離。2 次元空間での通常の距離計算。

$$D_{euc}(a,b) = \sqrt{\sum_{i=i}^{n}(a_i - b_i)^2}$$

   aとbはn次元ベクトル。
3. マンハッタン距離
   マンハッタン距離は 2 つの点間の格子状の道を歩く距離。都市のブロックを移動する距離のように各次元に沿って距離を測る。

$$D_{man}(a,b) = \sum_{i=i}^{n}|a_i - b_i|$$

4. Lp 距離
   これは一般化された距離測定方法であり、ユークリッド距離、マンハッタン距離の特殊なケースを含む。

$$D_{Lp}(a,b) = \left(\sum_{i=i}^{n}|a_i - b_i|^p\right)^{1/p}$$

   $p = 1$のとき、$D_{L1}(a,b)$は、マンハッタン距離
   $p = 2$のとき、$D_{L2}(a,b)$は、ユークリッド距離
5. マハラノビス距離
   データの分布を考慮して 2 つの点間の距離を計算する。データの共分散行列を使用して異なるスケールや相関の影響を除外する。

$$D_{mah}(a,b) = \sqrt{(a-b)^T S^{-1}(a-b)}$$

   ここでSはデータの共分散行列

2. 機械学習の分類

2.1. 教師あり学習

正解となる答えが含まれるデータをモデルに学習させる方法のこと。最終的な目標は正解ラベルのないデータを正解させること。

- 分類
  テストデータを離散的なカテゴリに分類する
  例）k-nn、サポートベクターマシン、ロジスティック回帰
- 回帰
  答えは連続した数値になる
  例）線形回帰、非線形回帰

2.2. 教師なし学習

与えられたデータの本質的な構造や法則をアルゴリズムが自動的に抽出するもの。

- クラスタリング
  データの中から特徴の似ているデータをグループごとに分けるタスク。
  例）k-means
- 次元削減
  データから重要な情報だけを抜き出し、あまり重要でない情報を削減するタスク。
  例）主成分分析、オートエンコーダ

2.3. 線形回帰

説明変数X、目的変数Yのような 2 次元のデータセットにおいては、XY 平面のデータ点に最もよく沿うような直線が描ける。この直線の式$y = ax + b$に予測したいデータXを代入し、予測値Yを計算する手法のこと。説明変数が 2 以上になると重回帰と呼ばれ、説明変数が 2 のとき数学的には 3 次元空間のもっとも当てはまりの良い平面を探索する。これらの探索には最小二乗法が用いられる。

$$y = X\omega + \epsilon$$

$$\begin{aligned} X &= (x_1, \ldots, x_n)^T && \text{説明変数} \\ y &= (y_1, \ldots, y_n)^T && \text{目的変数} \\ x_i &= (1, x_1)^T && \text{単回帰の場合} \\ \epsilon &= (\epsilon_1, \ldots, \epsilon_n)^T && \text{誤差関数} \\ \omega &= (\omega_0, \omega_1)^T && \omega_0\text{は切片、}\omega_1\text{は傾き} \end{aligned}$$

最小二乗法では平均二乗誤差の値を最小にするようなパラメータωを求める。平均二乗誤差は下に凸な二次関数であるので、微分したときに 0 となるωを求めればよい。

$$
\begin{aligned}
\frac{\partial}{\partial \omega}\mathrm{MSE} &= \frac{\partial}{\partial \omega}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2\right\} = 0 \\
\frac{\partial}{\partial \omega}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2\right\} &= \frac{\partial}{\partial \omega}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-X\omega)^2\right\} \\
&= \frac{1}{n}\frac{\partial}{\partial \omega}\left\{(y_i-X\omega)^T(y_i-X\omega)\right\} \\
&= \frac{1}{n}\frac{\partial}{\partial \omega}\left\{y_i^Ty_i-2\omega^TX^Ty_i+(X\omega)^TX\omega\right\} \\
&= \frac{1}{n}\left(2X^TX\omega-2X^Ty_i\right) \\
\therefore \omega &= (X^TX)^{-1}X^Ty_i
\end{aligned}
$$

ここで

$$
y_i^TX\omega = (\omega^TX^Ty_i)^T = \omega^TX^Ty_i \qquad \because \omega^TX^Ty_i\text{はスカラー} \tag{1}
$$

$$
\frac{\partial}{\partial x}x^Ta = a \tag{2}
$$

$$
\frac{\partial}{\partial x}x^TAx = 2Ax \tag{3}
$$

を計算で使った。

2.3. ハンズオン

~~ボストン住宅データセットを線形回帰で予測する。~~

2.4. 非線形回帰

　線形回帰ではデータの線形性を仮定していたが、データによっては非線形なものもある。通常の線形回帰では入力データをそのまま使用し、既定関数として恒等関数$f(x)=x$を使っていた。これに対して非線形なデータを表現するために多項式などの関数を用いることにより複雑な曲線を表現することができる。

基底関数の例

- 多項式基底関数

$$
\phi(x) = [1, x, x^2, ..., x^d]
$$

- ガウス基底関数

$$\phi(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- シグモイド基底関数

$$\phi(x) = \frac{1}{1+\exp(-a(x-b))}$$

2.5. ロジスティック回帰

　教師あり学習の一種で分類タスクに利用される。線形結合部分は線形回帰と同様だが、シグモイド関数の入力とすることで出力が最小 0 最大 1 の範囲に収まる。これはある事象が特定のクラスに属する確率と解釈できる。

$$z = W^T x + b$$
$$\sigma(z) = \frac{1}{1+e^{-az}}$$

シグモイド関数の微分は以下の通り

$$\begin{aligned}\frac{\partial \sigma(x)}{\partial x} &= \frac{\partial}{\partial x}\frac{1}{1+e^{-ax}} \\ &= \frac{-1}{(1+e^{-ax})^2}\cdot -ae^{-ax} \\ &= \frac{a}{1+e^{-ax}}\cdot\frac{1+e^{-ax}-1}{1+e^{-ax}} \\ &= a\frac{1}{1+e^{-ax}}\cdot\left(1-\frac{1}{1+e^{-ax}}\right) \quad = a\sigma(x)(1-\sigma(x))\end{aligned}$$

パラメータWは最尤推定法で探索する。

$$\text{尤度関数} \qquad L(W) = \prod_{i=1}^{N} P(y_i|x_i;W)$$
$$\because P(y_i=1|x_i;W) = \sigma(W^T x_i)$$
$$P(y_i=0|x_i;W) = 1-\sigma(W^T x_i)$$

$$
\begin{aligned}
\therefore L(W) &= \prod_{i=1}^{N} \sigma(W^T x_i)^{y_i} (1 - \sigma(W^T x_i))^{1-y_i} \\
-\log L(W) &= -\sum_{i=1}^{N} \left(y_i \log \sigma(W^T x_i) + (1 - y_i) \log(1 - \sigma(W^T x_i)) \right) \\
-\log L(W) &= -\sum_{i=1}^{N} \left(y_i \log P_i + (1 - y_i) \log(1 - P_i) \right) \\
&= E(W)
\end{aligned}
$$

対数尤度関数$E(W)$を最大化するために、勾配降下法を用いてパラメータWを更新する。

$$
\begin{aligned}
W &\Leftarrow W - \eta \frac{\partial E(W)}{\partial W} \\
\frac{\partial E(W)}{\partial W} &= \sum_{i=1}^{N} \frac{\partial E_i(W)}{\partial P_i} \frac{\partial P_i}{\partial W} \\
&= -\sum_{i=1}^{N} \left(\frac{y_i}{P_i} - \frac{1 - y_i}{1 - P_i} \right) P_i (1 - P_i) x_i \\
&= -\sum_{i=1}^{N} (y_i - P_i) x_i
\end{aligned}
$$

ここでηは学習率と呼ばれこの値が大きいと学習が発散する。繰り返し訓練することでパラメータが更新されていきある値で収束する。勾配降下法ではデータセットが大きくなるとメモリに載せられなくなるため確率的勾配降下法を使う。

2.5. ハンズオン

タイタニックの乗客データを利用しロジスティック回帰モデルを作る。

```
# %%
import pandas as pd
import plotly.express as px
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from IPython.display import display
# %%
train_df = pd.read_csv(r"study_ai_ml_google\data\titanic_train.csv")
test_df = pd.read_csv(r"study_ai_ml_google\data\titanic_test.csv")
display(train_df.head(5))
display(test_df.head(5))
# %%


def preprocess(data: pd.DataFrame) -> pd.DataFrame:
    replace_dict = {"male": 0, "female": 1}
    df = data[["Sex", "Age"]].copy()
    df["Sex"] = df["Sex"].apply(lambda x: replace_dict[x])
    df["Age"] = df["Age"].fillna(df["Age"].median(numeric_only=True))

    return df


display(train_df.describe())
display(train_df.isnull().sum())
label_data = train_df[["Survived"]].copy()
train_data = preprocess(train_df)
display(train_data.isnull().sum())
# %%
X_train, X_test, y_train, y_test = train_test_split(
    train_data, label_data, test_size=0.2, random_state=37
)
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('logreg', LogisticRegression())
])
pipeline.fit(X_train, y_train.values.ravel())
y_pred = pipeline.predict(X_test)
# %%
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
logreg = pipeline.named_steps['logreg']
coefficients = logreg.coef_
intercept = logreg.intercept_
print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

結果

```
Accuracy: 0.7877094972067039
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.85      0.82       101
           1       0.79      0.71      0.74        78

    accuracy                           0.79       179
   macro avg       0.79      0.78      0.78       179
weighted avg       0.79      0.79      0.79       179

Coefficients: [[ 1.16600323 -0.04096572]]
Intercept: [-0.63883289]
```

年齢 30 歳男性の予測値を計算すると、生存確率は約 19%になった。

```
predict [0]
prova   [[0.81491297 0.18508703]]
```

3. 機械学習の課題

3.1. 未学習と過学習

　未学習とは十分に学習が行われていないことで学習データに対する予測の精度が低い状態。過学習とは学習データに対する精度向上を重視しすぎる結果未知のデータに対する精度が下がってしまっている状態。

対策

- 未学習:表現力の高いモデルにする
- 過学習:不要な基底関数の削除、正則化項の利用などでモデルの表現力を下げる。学習データを増やす

3.2. 正則化

　過学習を抑制することを目的として、回帰係数が大きいことによるペナルティを与える罰則化項（正則化項）を導入する。

- L1 正則化
  回帰係数の絶対値の和を基準とする。スパース性をもつ。

LASSO 回帰

- L2 正則化
  回帰係数の 2 乗和を基準とする。
  Ridge 回帰

4. 検証集合

4.1. 汎化性能

　予測モデルは未知のデータに対してもよい当てはまりを示してほしい。そのため性能評価には学習に使ったデータとは別に検証用のデータを用意する。

4.2. ホールドアウト法

　データを学習用とテスト用の二つにある割合で分割し予測精度を評価する手法のこと。テストデータに偏りがある場合や、データセットが小さいと評価結果が偏ってものになる可能性がある。

4.3. クロスバリデーション

　データセットを複数のグループに分割し、それぞれのグループを一度ずつテスト用データとして使用し、残りのグループを学習用データとして使用する手法のこと。データセットを k 個のフォールドに分割し、k 回の訓練と評価を行う。それぞれの評価結果を平均することで予測精度を算出する。全データを複数回利用することで、より安定した予測精度の評価が得られるが計算コストが増える。

5. 性能指標

5.1. 回帰モデルの誤差集計方法

- R^2（決定係数）
  相関係数の二乗値。予測誤差を正規化すると得られる指標。
- $RMSE$（平方平均二乗誤差）
  予測誤差の二乗の平均。外れ値があるとよい評価ができない。
- MAE（平均絶対誤差）
  予測誤差の絶対値の平均。

5.2. 分類モデルの性能評価指標

- 混合行列

	positive（予測）	negative（予測）
positive（実際）	TruePositive	FalsePositive
negative（実際）	FalseNegative	TrueNegative

※正解ラベルが 1 のレコードを 0 と予測した回数 -> FalsePositive

- Accuracy
  正解率 $= \frac{TP+TN}{TP+FP+FN+TN}$
- 再現率
  再現率 $= \frac{TP}{TP+FN}$
- 適合率
  適合率 $= \frac{TP}{TP+FP}$
- F 値
  再現率と適合率の調和平均
  F値 $= \frac{2 \times 再現率 \times 適合率}{再現率+適合率}$

多クラス分類タスクではマクロ平均を使う。