判別分析

分析の評価

村田 昇

講義概要

- 第 1 回 : 判別分析の考え方
- **第 2 回 : 分析の評価**

判別分析の復習

判別分析

- 個体の特徴量からその個体の属するクラスを予測する関係式を構成
- **事前確率** : $\pi_k = P(Y = k)$ (prior probability)
  - $X = \boldsymbol{x}$ が与えられる前に予測されるクラス
- **事後確率** : $p_k(\boldsymbol{x})$ (posterior probability)
  - $X = \boldsymbol{x}$ が与えられた後に予測されるクラス

$$p_k(\boldsymbol{x}) = P(Y = k|X = \boldsymbol{x})$$

  - 所属する確率が最も高いクラスに個体を分類

判別関数

- 判別の手続き
  - 説明変数 $X = \boldsymbol{x}$ の取得
  - 事後確率 $p_k(\boldsymbol{x})$ の計算
  - 事後確率最大のクラスにデータを分類
- **判別関数** : $\delta_k(\boldsymbol{x})$ $(k = 1, \ldots, K)$

$$p_k(\boldsymbol{x}) < p_l(\boldsymbol{x}) \Leftrightarrow \delta_k(\boldsymbol{x}) < \delta_l(\boldsymbol{x})$$

  事後確率の順序を保存する計算しやすい関数
- 判別関数 $\delta_k(\boldsymbol{x})$ を最大化するようなクラス k に分類

線形判別

- $f_k(\boldsymbol{x})$ の仮定
  - q 変量正規分布の密度関数
  - 平均ベクトル $\boldsymbol{\mu}_k$: クラスごとに異なる
  - 共分散行列 Σ : **すべてのクラスで共通**

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{q/2}\sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right)$$

- 線形判別関数 : $\boldsymbol{x}$ の 1 次式

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

2 次判別

- $f_k(\boldsymbol{x})$ の仮定
  - q 変量正規分布の密度関数
  - 平均ベクトル $\boldsymbol{\mu}_k$: クラスごとに異なる
  - 共分散行列 Σ_k : **クラスごとに異なる**

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{q/2}\sqrt{\det \Sigma_k}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right)$$

- 2 次判別関数 : $\boldsymbol{x}$ の 2 次式

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\det \Sigma_k - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log \pi_k$$

Fisher の線形判別

- 新しい特徴量 $Z = \boldsymbol{\alpha}^\top X$ を考える
- 良い Z の基準
  - クラス内では集まっているほど良い ($\boldsymbol{\alpha}^\top W \boldsymbol{\alpha}$ は小)
  - クラス間では離れているほど良い ($\boldsymbol{\alpha}^\top B \boldsymbol{\alpha}$ は大)
- Fisher の基準

$$\text{maximize} \quad \boldsymbol{\alpha}^\top B \boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha}^\top W \boldsymbol{\alpha} = \text{const.}$$

  - $\boldsymbol{\alpha}$ は $W^{-1}B$ の第 1 から第 $K-1$ 固有ベクトル
  - 判別方法: 特徴量の距離を用いる
  - $d_k = \sum_{l=1}^{K-1}(\alpha_l^\top \boldsymbol{x} - \alpha_l^\top \mu_k)^2$ が最小のとなるクラス k に判別

2 値判別分析の評価

誤り率

- 単純な誤り

$$(\text{誤り率}) = \frac{(\text{誤って判別されたデータ数})}{(\text{全データ数})}$$

- 判別したいラベル : 陽性 (positive)
  - **真陽性** : 正しく陽性と判定 (true positive; TP)
  - **偽陽性** : 誤って陽性と判定 (false positive; FP) (**第 I 種過誤**)
  - **偽陰性** : 誤って陰性と判定 (false negative; FN) (**第 II 種過誤**)
  - **真陰性** : 正しく陰性と判定 (true negative; TN)

混同行列

	真値は陽性	真値は陰性
判別は陽性	真陽性 (True Positive)	偽陽性 (False Positive)
判別は陰性	偽陰性 (False Negative)	真陰性 (True Negative)

- **confusion matrix**
- 各条件にあてはまるデータ数を記載
- 転置で書く流儀もあるので注意 (次頁)

混同行列 (転置したもの)

	判別は陽性	判別は陰性
真値は陽性	真陽性 (True Positive)	偽陰性 (False Negative)
真値は陰性	偽陽性 (False Positive)	真陰性 (True Negative)

- パターン認識や機械学習で多く見られた書き方
- 誤差行列 (error matrix) とも呼ばれる

基本的な評価基準

- 定義

$$(\text{真陽性率}) = \frac{TP}{TP+FN} \qquad \text{(true positive rate)}$$
$$(\text{真陰性率}) = \frac{TN}{FP+TN} \qquad \text{(true negative rate)}$$
$$(\text{適合率}) = \frac{TP}{TP+FP} \qquad \text{(precision)}$$
$$(\text{正答率}) = \frac{TP+TN}{TP+FP+TN+FN} \qquad \text{(accuracy)}$$

- 別名 (分野で異なるので注意)
  - 感度 (sensitivity) あるいは 再現率 (recall)

$$(\text{真陽性率}) = \frac{TP}{TP+FN}$$

– 特異度 (specificity)

$$(\text{真陰性率}) = \frac{TN}{FP + TN}$$

– 精度 (accuracy)

$$(\text{正答率}) = \frac{TP + TN}{TP + FP + TN + FN}$$

F-値

- 定義 (**F-measure, F-score**)

$$F_1 = \frac{2}{1/(\text{再現率}) + 1/(\text{適合率})}$$
$$F_\beta = \frac{\beta^2 + 1}{\beta^2/(\text{再現率}) + 1/(\text{適合率})}$$

– 再現率 (真陽性率) と適合率の (重み付き) 調和平均

$$\text{調和平均} < \text{相乗平均} < \text{相加平均}$$

Cohen の kappa 値

- 定義 (Cohen's **kappa measure**)

$$p_o = \frac{TP + TN}{TP + FP + TN + FN} \qquad \text{(accuracy)}$$
$$p_e = \frac{TP + FP}{TP + FP + TN + FN} \cdot \frac{TP + FN}{TP + FP + TN + FN} + \frac{FN + TN}{TP + FP + TN + FN} \cdot \frac{FP + TN}{TP + FP + TN + FN}$$
$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}$$

– 観測された精度と偶然の精度の比較

受信者動作特性曲線

- **ROC 曲線** (receiver operating characteristic curve)
- 2 値判別関数 δ を用いた判定方法の一般形
  (c は事前確率に依存する項とも考えられる)

$$H(\boldsymbol{x}; c) = \begin{cases} \text{陽性}, & \delta(\boldsymbol{x}) > c \\ \text{陰性}, & \text{それ以外} \end{cases}$$

- 真陽性率と偽陽性率

$$\begin{aligned}\mathrm{TPR}(c) &= P(\text{陽性を正しく陽性と判別})\\ \mathrm{FPR}(c) &= P(\text{陰性を誤って陽性と判別})\\ &= 1 - P(\text{陰性を正しく陰性と判別})\end{aligned}$$

- **ROC 曲線** : $H(x;c)$ の c を自由に動かし x 軸に偽陽性率，y 軸に真陽性率を描画したもの
  - 一般に ROC 曲線は $(0,0)$ と $(1,1)$ を結ぶ右肩上りの曲線
  - 理想的な判別関数は $(0,1)$(完全な判別) を通る
  - 曲線と x 軸で囲まれた面積が広い $\Leftrightarrow$ 良い判別方法
- **AUC** : 上記の面積 (area under the ROC curve)
  - ROC 曲線を数量化する方法の一つ
  - 判別関数の良さ (2 値判別の難しさ) を測る基準の一つ

実習

データセットの準備

- 以下のデータセットを使用
  - `winequality-red.csv`

    UC Irvine Machine Learning Repository で公開されている Wine Quality Data Set の一部

    https://archive.ics.uci.edu/ml/datasets/Wine+Quality
  - 以下に download せずに読み込む方法を紹介する

```
wq_data <-
  read_delim("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red
             delim = ";") |> # 区切り文字が';'
  mutate(grade = factor(case_when( # quality を A,B,C,D に割り当てる
            quality >= 7 ~ "A",
            quality >= 6 ~ "B",
            quality >= 5 ~ "C",
            .default = "D")))
```

R：判別結果の評価

- 評価のための枠組
  - **caret** : Max Kuhn @Rstudio によるパッケージ
    - https://topepo.github.io/caret/
  - **tidymodels** : Max Kuhn, Hadley Wickham @RStudio による `tidyverse` 向けに再設計されたパッケージ
    - https://www.tidymodels.org
- 本講義では tidymodels を中心に説明
- パッケージ集の利用には以下が必要

```
#' 最初に一度だけ以下のいずれかを実行しておく
#'  - Package タブから tidymodels をインストール
#'  - コンソール上で次のコマンドを実行 'install.packages("tidymodels")'
#' tidymodels パッケージの読み込み
library(tidymodels)
```

R：混同行列

- 関数 yardstick::conf_mat()

```
conf_mat(data, truth, estimate,
  dnn = c("Prediction", "Truth"), case_weights = NULL, ...)
#' data: 真値と予測値が含まれるデータフレーム
#' trush: 真値（ラベル）の列名
#' estimate: 予測値（ラベル）の列名
#' 詳細は '?yardstick::conf_mat' を参照
```

- 集計や視覚化のために補助的な関数

```
#' object: conf_mat の出力
#' 様々な評価指標を tibble 形式で出力
#' 詳細は '?yardstick::summary.conf_mat' を参照
summary(object,
  prevalence = NULL, beta = 1, estimator = NULL,
  event_level = yardstick_event_level(), ...)
#' 混同行列を図示
autoplot(object, type = c("mosaic", "heatmap"))
```

R：ROC 曲線

- 関数 yardstick::roc_curve()

```
roc_curve(data, truth, ...,
  na_rm = TRUE, event_level = yardstick_event_level(),
  case_weights = NULL, options = list())
#' data: 真値と予測値が含まれるデータフレーム
#' trush: 真値（ラベル）の列名
#' ...: 予測値（ラベル）の事後確率を与える列名
#' 詳細は '?yardstick::roc_curve' を参照
```

- 視覚化のために補助的な関数

```
#' object: roc_curve の出力
#' ROC 曲線を図示
autoplot(object)
```

- 関数 yardstick::roc_auc()

```
roc_auc(data, truth, ..., estimator = NULL,
        na_rm = TRUE,  event_level = yardstick_event_level(),
        case_weights = NULL, options = list())
#' data: 真値と予測値が含まれるデータフレーム
#' trush: 真値（ラベル）の列名
#' ...: 予測値（ラベル）の事後確率を与える列名
#' 詳細は '?yardstick::roc_auc' を参照
```

練習問題

- 前回と同様に東京の気候データの線形判別を行い，以下を確認しなさい
  - 9 月と 10 月の気温と湿度のデータを抽出する

```
tw_data <- read_csv("data/tokyo_weather.csv")
tw_subset  <- tw_data |>
  filter(month %in% c(9,10)) |>
  select(temp, humid, month) |>
  mutate(month = as_factor(month)) # 月を因子化
```

- 全てデータを用いて線形判別関数を構成する
- 構成した判別関数の評価を行う
- ROC 曲線を描画し，AUC を求める

R：訓練・試験データの分割

- 関数 `rsample::initial_split()`

```
initial_split(data, prop = 3/4,
              strata = NULL, breaks = 4, pool = 0.1, ...)
#' data: データフレーム
#' prop: 訓練データの比率
#' strata: 層別に分割する場合の変数
#' 詳細は '?rsample::initial_split' を参照
```

- 訓練・試験データの取得のための関数

```
#' x : initial_split の出力
#' 訓練データの取得
training(x, ...)
#' 試験データの取得
testing(x, ...)
```

練習問題

- Wine Quality Data Set を用いて以下を確認しなさい
  - 8:2 の比率で訓練データと試験データに分割する
  - 訓練データを用いて線形・2 次判別関数を構成する
  - 訓練データを用いて評価を行う
  - 試験データを用いて評価を行う

予測誤差

訓練誤差と予測誤差

- **訓練誤差**：既知データに対する誤り (training error)
- **予測誤差**：未知データに対する誤り (predictive error)
- 訓練誤差は予測誤差より良くなることが多い
- 既知データの判別に特化している可能性がある
  - 過適応 (over-fitting)
  - 過学習 (over-training)
- 予測誤差が小さい ⇔ 良い判別方法

交叉検証

- データを訓練データと試験データに分割して用いる
  - **訓練データ**：判別関数を構成する (training data)
  - **試験データ**：予測精度を評価する (test data)
- データの分割に依存して予測誤差の評価が偏る
- 偏りを避けるために複数回分割を行ない評価する
- “交差” と書く場合もある

交叉検証法

- **cross-validation (CV)**
- k-重交叉検証法 (k-fold cross-validation; k-fold CV)
  - n 個のデータを k ブロックにランダムに分割
  - 第 i ブロックを除いた $k-1$ ブロックで判別関数を推定
  - 除いておいた第 i ブロックで予測誤差を評価
  - $i = 1, \ldots, k$ で繰り返し k 個の予測誤差で評価 (平均や分散)
- leave-one-out 法 (leave-one-out CV; LOO-CV)
  - $k = n$ として上記を実行

実習

R：LOO 交叉検証法

- 関数 `lda()` と `qda()` はオプションで LOO 交叉検証を行うことができる
- オプションの指定方法

```
toy_lda <- lda(formula, toy_data, CV = TRUE)
toy_lda[["class"]] # LOO CV による予測結果
#' 特定のデータを除いて判別関数を構成し，そのデータの予測を行っている
toy_qda <- qda(formula, toy_data, CV = TRUE)
toy_qda[["class"]] # LOO CV による予測結果
#' 2次判別についても同様
```

練習問題

- MASS::biopsy を用いて 2 次判別の分析を行いなさい
  - 全てのデータを用いて訓練誤差を評価する
  - LOO 交叉検証法を用いて予測誤差を評価する

R：k-重交叉検証法

- `tidymodels` パッケージの関数群を利用

```
#' 交叉検証用のデータ分割
#' 詳細は 'rsample::vfold_cv' を参照
vfold_cv(data, v = 10, repeats = 1,
         strata = NULL, breaks = 4, pool = 0.1, ...)
#' 最も簡単な処理の流れ (以下の関数の組わ合わせ)
#' 詳細は 'workflows::workflow'
#' および 'tune::fit_resamples' を参照
workflow() |>
  add_formula(目的変数 ~ 説明変数) |>
  add_model(推定に用いるモデル) |>
  fit_resamples(resamples = vfold_cv の出力)
#' 評価の取得
#' 詳細は 'tune::collect_metrics' を参照
collect_metrics(fit_resamples の出力)
```

練習問題

- Wine Quality Data Set を用いて線形判別と 2 次判別の分析を行いなさい
  - LOO 交叉検証法を用いて予測誤差を評価する
  - k-重交叉検証法を用いて予測誤差を評価する

次回の予定

- **第 1 回 : クラスタ分析の考え方と階層的方法**
- 第 2 回 : 非階層的方法と分析の評価