Project 1: Classification

Updates

- 2021-04-12 Fixed some spelling mistakes and removed misleading comment on cross-validation from question 5
- 2021-04-13 Added a hint to Question 1
- 2021-04-19
  - Fixed typo in Question 5 (*Use the metric from question 4, not from question 3*)
  - Added a recommendation in Question 5 on the size of training and test data
  - Added general note on allowed software under Introduction
  - Added note on how to compute correlation matrix in question 3
  - When using my code snippet for loading the data in R, it is advised to add `as.factor` before the `case_when` when the labels get translated from `B` and `M` to `0` and `1`. This makes some packages such as `randomForest` easier to run.
- 2021-04-21 Added some clarification to Question 4

Introduction

This project is all about classification algorithms. In the first four lectures you learned about purely data driven classification algorithms such as kNN, model-based algorithms such as discriminant analysis and logistic regression, as well as partition/tree-based models such as CART or Random Forest.

In this project you will explore different properties of these algorithms.

Note about allowed software You can use established packages, e.g. `scikit-learn` in Python or `caret` in R, and do not have to implement algorithms on your own.

Note about submission

Only questions 2, 4, and 5 are supposed to be part of the presentation you submit.

Part 1: Geometry of the feature space

The first crucial thing to internalize with all classification algorithms is that they make certain assumptions about how features are distributed and how features and classes are connected.

Question 1

Go back through the lectures and answer the following questions:

What assumptions do the following methods make about the distribution of features for a certain class? What does this mean abstractly but also visually?

$$p(x|i) = \frac{p(i|x)\,p(x)}{\sum_x p(i|x)\,p(x)}$$

1. Linear discriminant analysis
2. Quadratic discriminant analysis
3. kNN
4. CART

Also think about the impact of k in kNN and the impact of different parameters on the CART model.

Hint Think about how each algorithm performs the classification. This tells you a lot about how features and classes are related. In more abstract terms (if that is your thing), think not just about a single $\boxed{p(\mathbf{x}|i)}$ but also think about how these distributions are connected with the resulting classification rule $c(\mathbf{x})$. For example, nearest centroids classification assumes that $p(\mathbf{x}|i) = N(\mathbf{x}; \mu_i, \sigma^2\mathbf{I})$ and then assigns classes using the decision rule

$$c(\mathbf{x}) = \operatorname*{argmin}_i \|\mathbf{x} - \mu_i\|_2.$$

So if one were to simulate data that is perfect for nearest centroids, then one would fix centers, choose a σ and simulate feature vectors from the distributions $p(\mathbf{x}|i)$. Now comes a crucial step: Set the class labels using the classification rule, instead of setting the class label of each sample to the distribution it was simulated from (i.e. setting y to i if $\mathbf{x}$ came from $p(\mathbf{x}|i)$). This way, the data is simulated exactly as nearest centroids assumes it to be.

Generalise this thinking to the methods mentioned above.

Question 2

For the following questions focus on QDA and CART.

1. Simulate training and test data that follows the assumptions for each of the two models. (QDA, CART)

2. What are your expectations on the test error if you were to apply QDA to data simulated for CART and *vice versa*?
3. Train the methods on both training data sets and compare their respective test errors for each dataset. Were your expectations from Step 2 correct? Any surprises? **Discuss**!
4. Taking what you have learned from the project so far, can you simulate data for which the target method works best? (*Clarifying example:* Data simulated for QDA should be classified best by QDA and less well CART, and the other way around)

Note Make sure you repeat the simulation of the data many times. A result based on a single training and test dataset could be pure chance, especially due to the instability of CART. Write your code in such a fashion that you can easily create many training and test datasets repeat the comparison described above. **Always report average values but always with a measure of uncertainty (e.g. standard deviation).**

For most practical purposes you would not use a single CART model, but rather a bagged version or Random Forests. This exercise is meant to give you a feeling of the different capabilities of the two models.

Part 2: Actual data and classification metrics

For this question we will work with the UCI breast cancer dataset.

In [ ]:

```python
import pandas as pd

# Load UCI breast cancer dataset with column names and remove ID column
uci_bc_data = pd.read_csv(
    "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data",
    sep=",",
    header=None,
    names=[
        "id_number", "diagnosis", "radius_mean",
        "texture_mean", "perimeter_mean", "area_mean",
        "smoothness_mean", "compactness_mean",
        "concavity_mean","concave_points_mean",
        "symmetry_mean", "fractal_dimension_mean",
        "radius_se", "texture_se", "perimeter_se",
        "area_se", "smoothness_se", "compactness_se",
        "concavity_se", "concave_points_se",
        "symmetry_se", "fractal_dimension_se",
        "radius_worst", "texture_worst",
        "perimeter_worst", "area_worst",
        "smoothness_worst", "compactness_worst",
        "concavity_worst", "concave_points_worst",
        "symmetry_worst", "fractal_dimension_worst"
    ],).drop("id_number", axis=1)
```

```
y = uci_bc_data.diagnosis.map({"B": 0, "M": 1}).to_numpy()
X = uci_bc_data.drop("diagnosis", axis=1).to_numpy()
```

Classes are either *benign*, which is denoted as B and stands for a "harmless" tumour, and *malignant*, which is denoted as M and stands for a life-threatening tumour. Clearly malignant tumours are more dangerous and it is more important to detect these. Normally, the class of major interest is coded as 1 and the class of less interest is coded as 0.

In [ ]:

```
## The following R code loads the data in the same fashion as the Python code above

library(readr)
library(tibble)
library(dplyr)
library(purrr)

names <- c("id_number", "diagnosis", "radius_mean",
           "texture_mean", "perimeter_mean", "area_mean",
           "smoothness_mean", "compactness_mean",
           "concavity_mean","concave_points_mean",
           "symmetry_mean", "fractal_dimension_mean",
           "radius_se", "texture_se", "perimeter_se",
           "area_se", "smoothness_se", "compactness_se",
           "concavity_se", "concave_points_se",
           "symmetry_se", "fractal_dimension_se",
           "radius_worst", "texture_worst",
           "perimeter_worst", "area_worst",
           "smoothness_worst", "compactness_worst",
           "concavity_worst", "concave_points_worst",
           "symmetry_worst", "fractal_dimension_worst")

uci_bc_data <- read_delim(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data",
  delim = ",",
  col_names = names,
  col_types = cols(
    .default = col_number(),
    id_number = col_integer(),
    diagnosis = col_factor()
  ))

y <- uci_bc_data %>%
  mutate(diagnosis = as.factor(case_when(diagnosis == "B" ~ 0, diagnosis == "M" ~ 1))) %>%
  select(diagnosis) %>%
  as_vector() %>%
  unname()
X <- uci_bc_data %>%
  select(-id_number, -diagnosis) %>%
  as.matrix()
```

When starting to investigate a classification dataset, there should be two basic steps you always (and especially now) take.

1. Investigate if the class labels are balanced or imbalanced.
2. Investigate the features
   A. Are they numerical or categorical?
   B. Do they have highly varying scales? Should data be scaled/normalized/centred before being used in a classification method? (More on this in lecture 5)
   C. How does the correlation matrix between the features look like? Are there highly correlated features? Are there plausible reasons for the correlations?

Question 3

First perform the two investigations above and then train a Quadratic Discriminant Analysis model on the data and compare the **correlation matrix** of the first 10 features in the data to the **correlation matrices** (of the first 10 features) for classes 0 and 1 estimated for the QDA model.

Hint 1 When using Python and `scikit-learn` you have to set `store_covariance=True` when creating the `QuadraticDiscriminantAnalysis` object. After fitting the classifier the covariance matrices are stored in `fit.covariance_`.

If you are using R and the `MASS` package to compute the QDA model, then the covariance matrix of class i can be computed as

```
solve(fit$scaling[,,i] %*% t(fit$scaling[,,i]))
```

Hint 2 Once you have a covariance matrix $\mathbf{\Sigma}$ you can compute the correlation matrix $\mathbf{C}$ in the following way:

Let $\mathrm{diag}(\mathbf{\Sigma})$ be the diagonal entries of the covariance matrix $\mathbf{\Sigma}$ as a vector, and let $\mathrm{Diag}(\mathbf{v})$ a diagonal matrix with the vector $\mathbf{v}$ on its diagonal. Set $\mathbf{D} = \mathrm{Diag}(\sqrt{1/\mathrm{diag}(\mathbf{\Sigma})})$, where the inverse and square root are interpreted elementwise. Then the correlation matrix is obtained as

$$\mathbf{C} = \mathbf{D}\mathbf{\Sigma}\mathbf{D}.$$

Question 4

As mentioned above, it is more important for us to discover malignant tumours.

1. Choose a classification methods (classifier) and test the effect of different classification metrics (such as accuracy, specificity, the F1 score, etc.). What I mean by this is the following
   - Split your data into a bunch of folds (do this without stratification for now)
   - For each fold $\mathcal{F}$:
     - Train a model on the remaining training folds using the models standard mode of training (e.g. optimisation of a likelihood in the case of QDA, splitting on the Gini score in CART or Random Forest)
     - Compute classification metrics on the test fold $\mathcal{F}$
   - Report the average performance across folds
2. Which classification metric(s?) is/are most suitable for our goal here? **Explain why**!
3. Choose one additional classification method that is substantially different (i.e. QDA and Random Forest, or logistic regression and CART. Not QDA and LDA, or logistic regression and LDA, they are too similar in their assumptions) and compare the two methods using the best classification metric(s) you determined in Step 2.
4. Does using stratified cross-validation change/improve your results?

Mislabeling in training data

So far, we assumed that the class labels in training data for a classification problem were correct. What happens if observations are randomly or systematically mislabelled? In a real life situation this could happen if e.g. the person collecting the data confuses the class labels (*random mislabelling*) or if a machine was programmed wrongly and consistently returns the wrong label (*systematic mislabelling*).

Question 5

Use the dataset you used for the previous question and simulate *random mislabelling* for p percent of the data, i.e. randomly select p percent of the samples and flip their class label.

Perform a small **simulation study**.

1. Choose a classification method and a sequence of at least 10 values for p between 0 and 100 percent.
2. Set aside a proportion of the data as a test set (think about stratification!) and use the rest as training data (e.g. 75% of the data for training and 25% of the data for testing). Assume that mislabeling only appears in the training data but not in the test data.
3. Repeatedly simulate new mislabelings of the training data for the current value of p and collect results in terms of the best metric you identified in Question 4. Evaluate that metric on the test data that you set aside.

For better stability of the result it is important to also repeat the splitting of the data a few times.

Goal Analyse the results in terms of p. How much mislabeling can the algorithm withstand? By this I mean: How stable in terms of the classification metric is the classifier when the percentage of mislabeling increases.

Hint A reasonable number of repeats are 5-10 for the train/test split, 20-100 training data mislabeling

Bonus question: Do you get different results if **only** those samples labelled 0 become occasionally 1 or only those samples labelled 1 become 0?