Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Supervised learning: Classification

- Input $X = (X_1, X_2, \ldots, X_d)$ is a vector of d categorical variables representing **features** of an individual or item
- Output Y is a categorical variable denoting a **class label**
- Goal: predict Y given X and **training data** of n individuals or items

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$$

- Features are stored in a **data matrix** and class labels in a vector

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Medical diagnosis using classification

- In the medical diagnosis of breast cancer problem, the **class label** of interest is

$$\{Y = 0\} = \{\text{a woman having no breast cancer}\}$$
$$\{Y = 1\} = \{\text{a woman having breast cancer}\}$$

- The **feature** here is the outcome of a mammogram test

$$\{X = 0\} = \{\text{mammogram test is negative}\}$$
$$\{X = 1\} = \{\text{mammogram test is positive}\}$$

- **Training data**

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \in \{0, 1\} \times \{0, 1\}$$

contains the mammogram outcome and breast cancer status of n women

Medical diagnosis using classification

- We will build a joint model for X and Y using

$$P(X = x, Y = y) = P(Y = y)P(X = x|Y = y)$$

- The distribution on the cancer status

$$P(Y = y), \qquad y \in \{0, 1\}$$

represents our **prior beliefs** about breast cancer without mammogram testing

- The conditional distribution

$$P(X = x|Y = 1), \quad x \in \{0, 1\}$$

models how likely the mammogram results in a **true positive**

- The conditional distribution

$$P(X = x|Y = 0), \quad x \in \{0, 1\}$$

models how likely the mammogram results in a **false positive**

Medical diagnosis using classification

- Using Bayes' theorem, we have a **Bayes classifier**

$$P(Y = y|X = x) = \frac{P(Y = y)P(X = x|Y = y)}{P(X = x)}$$

- The medical diagnosis task is based on the probabilities

$$P(Y = 1|X = 0) \qquad \text{and} \qquad P(Y = 1|X = 1)$$

Medical diagnosis in R

- The dataset stored as `breastcancer.RData` and save it in a folder dedicated to this course

- In RStudio, click on "Set Working Directory" > "Choose Directory..." and select folder

- This dataset of $n = 100,000$ individuals contains:
  1. `cancer_status`: breast cancer status
  2. `mammogram_outcome`: outcome of mammogram test

Medical diagnosis in R

- Given the training data

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \in \{0,1\} \times \{0,1\},$$

how can we learn $P(Y = y), y \in \{0,1\}$?

- We can simply use the sample proportion

$$\hat{P}(Y = 0) = \frac{1}{n}\sum_{i=1}^{n}(1 - y_i), \qquad \hat{P}(Y = 1) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

- We can implement this using the `mean` function

```
prob_cancer <- mean(cancer_status)
```

Medical diagnosis in R

- Given the training data

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \in \{0,1\} \times \{0,1\},$$

  how can we learn $P(X = x | Y = 1), x \in \{0,1\}$?

- Since we are only interested in $Y = 1$, we subset the training data to cases of woman with cancer

```
outcome_cancer <- mammogram_outcome[cancer_status == 1]
```

- We can then use the sample proportion of the subsetted data

$$\hat{P}(X = 1 | Y = 1) = \frac{\sum_{i=1}^{n} y_i \mathbf{x}_i}{\sum_{i=1}^{n} y_i}$$

- We can implement this using the `mean` function

```
prob_positive_cancer <- mean(outcome_cancer)
```

Medical diagnosis in R

- Similarly, to learn $P(X = x|Y = 0), x \in \{0, 1\}$, we subset the training data to cases of woman without cancer

```
outcome_nocancer <- mammogram_outcome[cancer_status == 0]
```

- We can then use the sample proportion of the subsetted data

$$\hat{P}(X = 1|Y = 0) = \frac{\sum_{i=1}^{n}(1 - y_i)\mathbf{x}_i}{\sum_{i=1}^{n}(1 - y_i)}$$

- We can implement this using the `mean` function

```
prob_positive_nocancer <- mean(outcome_nocancer)
```

Medical diagnosis in R

- After fitting the **joint model**

$$\hat{P}(X = x, Y = y) = \hat{P}(Y = y)\hat{P}(X = x|Y = y)$$

we can compute the **marginal probability** using the law of total probability

$$\begin{aligned}\hat{P}(X = 1) &= \hat{P}(X = 1, Y = 0) + \hat{P}(X = 1, Y = 1) \\ &= \hat{P}(Y = 0)\hat{P}(X = 1|Y = 0) \\ &+ \hat{P}(Y = 1)\hat{P}(X = 1|Y = 1)\end{aligned}$$

- We implement this as follows

```
prob_positive <- (1-prob_cancer) * prob_positive_nocancer +
                 prob_cancer * prob_positive_cancer
```

Medical diagnosis in R

- We can now compute the desired probability

$$\hat{P}(Y = 1|X = 1) = \frac{\hat{P}(Y = 1)\hat{P}(X = 1|Y = 1)}{\hat{P}(X = 1)}$$

- We implement this as follows

```
prob_cancer_positive <- prob_cancer*prob_positive_cancer /
                        prob_positive
```

- Exercise: compute $\hat{P}(Y = 1|X = 0)$

Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Bayes classifiers

- We will build a joint model for X and Y using

$$P(X = x, Y = y) = P(Y = y)P(X = x|Y = y)$$

where the values x and y are not necessarily binary

- The **prior distribution** on the classes

$$P(Y = y)$$

represents our beliefs about Y without knowledge of X

- The **class conditional distribution** of the features

$$P(X = x|Y = y) = P(X_1 = x_1, \ldots, X_d = x_d|Y = y)$$

models the features in each class

- Using Bayes' theorem, we have a **Bayes classifier**

$$P(Y = y|X = x) = \frac{P(Y = y)P(X = x|Y = y)}{P(X = x)}$$

where $P(X = x)$ is the **marginal distribution** of the features

Naive Bayes classifiers

- To simplify the Bayes classifier, it is common to assume that the features $X_1, \ldots, X_d$ are independent given each class

$$P(X = x|Y = y) = P(X_1 = x_1, \ldots, X_d = x_d|Y = y)$$
$$= \prod_{i=1}^{d} P(X_i = x_i|Y = y)$$

- Such an independence assumption is known as **conditional independence** and is not the same as having **marginal independence**

$$P(X = x) = P(X_1 = x_1, \ldots, X_d = x_d)$$
$$= \prod_{i=1}^{d} P(X_i = x_i)$$

- Under conditional independence of the features, the classifier is known as a **naive Bayes classifier**

Naive Bayes classifiers

- Given the training data

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n),$$

  how can we learn $P(Y = y)$?
- Like before, we use the sample proportion of class label y

$$\hat{P}(Y = y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i = y)$$

  where the **indicator function** is

$$\mathbb{I}(y_i = y) = \begin{cases} 1, & \text{if } y_i = y \\ 0, & \text{otherwise} \end{cases}$$

- Implement indicator function in R using the **comparison operator ==**

Naive Bayes classifiers

- Given the training data

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n),$$

how can we learn $P(X_i = k | Y = y)$ for feature $i = 1, \ldots, d$?

- Like before, we use the sample proportion of each category k of feature i after subsetting the data to cases involving only class label y

$$\hat{P}(X_i = k | Y = y) = \frac{\sum_{j=1}^{n} \mathbb{I}(y_j = y)\mathbb{I}(x_{j,i} = k)}{\sum_{j=1}^{n} \mathbb{I}(y_j = y)},$$

- This fits the model

$$\hat{P}(X = x, Y = y) = \hat{P}(Y = y) \prod_{i=1}^{d} \hat{P}(X_i = x_i | Y = y)$$

which corresponds to the **maximum likelihood** approach to learn these unknown probabilities

Titanic dataset in R

- Using the `read.csv` function from the `utils` R package, read the dataset stored as `titanic.csv` into the global environment

```
library(utils)
dataset <- read.csv("titanic.csv", colClasses =
"factor")
```

- This dataset provides information on the fate of passengers on the fatal maiden voyage of the ocean liner Titanic
- There are 2201 datapoints with 4 variables
- Each row represents a passenger on Titanic
- Column variables:
  `Class`: 1st, 2nd, 3rd or Crew
  `Sex`: Male or Female
  `Age`: Child or Adult
  `Survived`: No or Yes

Naive Bayes classifiers in R

- We will predict the outcome variable $Y =$ Survived using $X_1 =$ Class, $X_2 =$ Sex and $X_3 =$ Age as features
- We expect these features to contain survival information:
  – "women and children" first policy
  – policy was not entirely successful in the third class
- To apply the naive Bayes classifier, we will use the `naiveBayes` function from the R package `e1071`
- Install the R package using

```
install.packages("e1071")
```

- Load the R package using

```
library(e1071)
```

Naive Bayes classifiers in R

- We fit the naive Bayes classifier using

```
model <- naiveBayes(Survived ~ Class + Sex + Age,
                    data = dataset)
```

- The first argument, `formula`, specifies our model
- The second argument `data` specifies the dataset
- The output of the function call is stored as `model`

Naive Bayes classifiers in R

- `model` contains the prior probabilities

$$\hat{P}(Y = \text{No}) \qquad \text{and} \qquad \hat{P}(Y = \text{Yes})$$

and the conditional probabilities

$$P(X_1 = k|Y = y), \quad k \in \{1\text{st}, 2\text{nd}, 3\text{rd}, \text{Crew}\}, \quad y \in \{\text{No}, \text{Yes}\},$$
$$P(X_2 = k|Y = y), \quad k \in \{\text{Male}, \text{Female}\}, \quad y \in \{\text{No}, \text{Yes}\},$$
$$P(X_3 = k|Y = y), \quad k \in \{\text{Child}, \text{Adult}\}, \quad y \in \{\text{No}, \text{Yes}\}$$

- Sanity check

```
survived <- dataset$Survived == "Yes"
mean(survived)
mean(dataset$Class[survived] == "Crew")
```

Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Prediction with naive Bayes classifiers

- After fitting the naive Bayes classifier, we can examine its **predictive accuracy**
- Given features $X_1 = \text{Class}$, $X_2 = \text{Sex}$ and $X_3 = \text{Age}$ of a passenger, we will compute the posterior probabilities using Bayes' theorem

$$P(Y = \text{No} \,|\, X_1 = \text{Class}, X_2 = \text{Sex}, X_3 = \text{Age})$$
$$P(Y = \text{Yes} \,|\, X_1 = \text{Class}, X_2 = \text{Sex}, X_3 = \text{Age})$$

- Our **prediction** $\hat{Y} \in \{\text{No}, \text{Yes}\}$ will be given by the most likely outcome
- Implement this **prediction rule** on our training dataset using

```
predictions <- predict(model, dataset)
```

Assessing predictive accuracy

- A **misclassification** happens if the prediction is not the actual outcome

- The **misclassification rate** on our training dataset is

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left(\hat{Y}_i \neq y_i\right)$$

  where $\hat{Y}_i$ is the prediction for $i = 1, \ldots, n$ datapoint $(\mathbf{x}_i, y_i)$

- Compute the **misclassification rate** on our training dataset

```
mean(predictions != dataset$Survived)
```

Assessing predictive accuracy

- To analyze the **misclassification error** further, we can do a crosstabulation/contingency table of our predictions against the actual outcomes

```
table(predictions, dataset$Survived)
```

- This table is known as the **confusion matrix**
- Depending on the context, if the two types of error are not equally severe, we may want to use a different **loss function** instead of the **misclassification rate**
- As we used the training dataset to both fit our model and assess predictive accuracy, this often underestimates the misclassification error

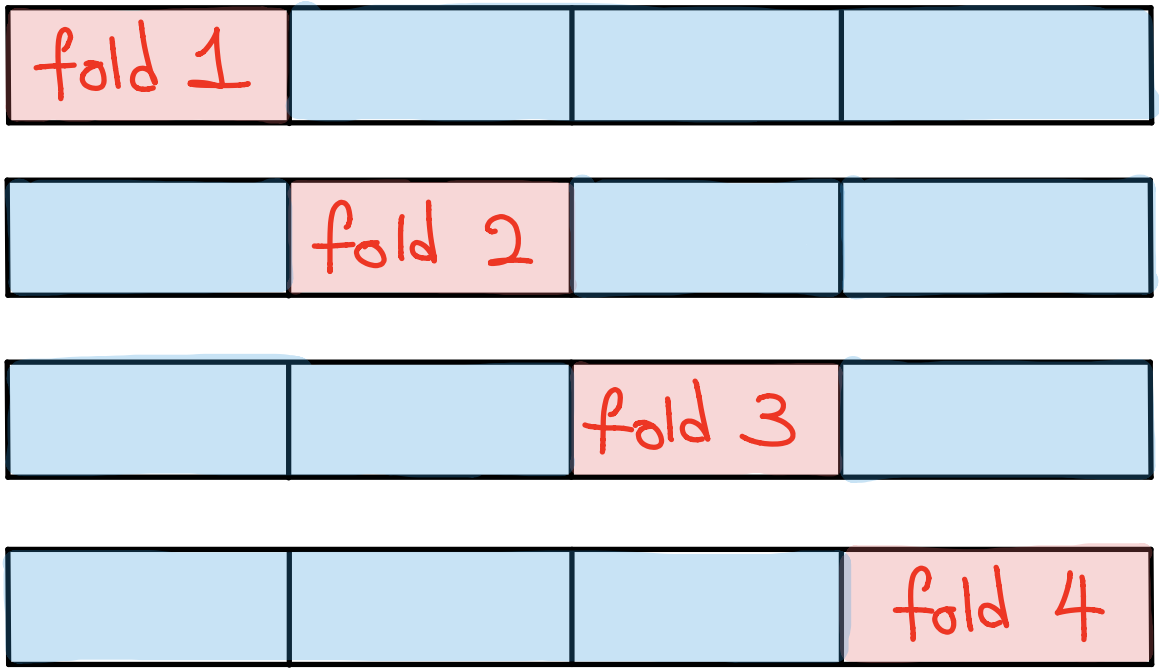
Cross validation

- In an ideal scenario, we would have a **training dataset** (to train our model) and a **testing dataset** (to test our predictions accuracy)
- Hardly the case in practice: we are only give one dataset

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$$

- Simple idea: split dataset into two blocks, one for training, one for validation
- This procedure is commonly used in practice due to its generality
- If dataset is small, this procedure will be unreliable as we will not have enough datapoints to train and test

K-fold cross validation



- A popular solution is K-**fold cross validation**
  - **split** dataset into K folds
  - **train** on all data except $k \in \{1, 2, \ldots, K\}$ fold
  - **test** on data in $k \in \{1, 2, \ldots, K\}$
  - **average** prediction error over folds

Leave one out cross validation

- n-fold cross validation is called **leave one out cross validation**
- We can implement this using a for loop over the training dataset

```
n <- nrow(dataset)
misclassification <- logical(n)
for (i in 1:n){
  model <- naiveBayes(formula = Survived ~ Class + Sex + Age,
                      data = dataset[-i,])
  predictions <- predict(model, dataset[i,])
  misclassification[i] <- predictions != dataset$Survived[i]
}
mean(misclassification)
```

US congressional voting records (1984) dataset in R

- Read the dataset stored as `HouseVotes84.csv` into the global environment
- This dataset contains votes for each of the 435 US House of Representatives Congressmen on 16 key votes
- There are 435 datapoints with 16 variables
- Each row represents a congressman or congresswoman
- Column variables (summarized):
  `V1`: handicapped infants (No or Yes)
  `V2`: water project cost sharing (No or Yes)
  ⋮ ⋮ ⋮ ⋮ ⋮
  `V16`: export administration act South Africa (No or Yes)
- More information in `http://archive.ics.uci.edu/ml/datasets/congressional+voting+records`

US congressional voting records (1984) dataset in R

- We will predict the party affiliation variable $Y =$ Party using $X_1 =$ Vote 1, $X_2 =$ Vote 2, $\ldots$, $X_{16} =$ Vote 16 as features
- We expect voting preference to contain strong information about party affiliation (democrat or republican)
- We use 75% of the dataset for training, and 25% for testing

```
n <- nrow(dataset)
m <- 326
xtrain <- dataset[1:m, -1]
xtest <- dataset[(m+1):n, -1]
ytrain <- dataset[1:m, 1]
ytest <- dataset[(m+1):n, 1]
```

US congressional voting records (1984) dataset in R

- We fit the naive Bayes classifier on the **training dataset**

```
model <- naiveBayes(x = xtrain, y = ytrain)
```

- Instead of specifying `formula`, we use the arguments
  `x`: data matrix of features
  `y`: class labels

- We perform prediction on the **testing dataset**

```
predictions <- predict(model, xtest)
```

- We assess prediction accuracy as before

```
mean(predictions != ytest)
table(predictions, ytest)
```

Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Naive Bayes classifiers in R

- By using the **maximum likelihood** approach to train naive Bayes classifiers, overfitting can occur in practice
- To prevent overfitting, we could use **Bayesian inference** to learn the unknown probabilities

$$\hat{P}(X = x, Y = y) = \hat{P}(Y = y) \prod_{i=1}^{d} \hat{P}(X_i = x_i | Y = y)$$

  defining the naive Bayes classifier
- You will implement an instance of this for your project
- Caution: do not confuse Bayesian inference (i.e. using Bayes' theorem to learn parameters of a model) with Bayes' classifier (i.e. using Bayes' theorem to predict Y given X)

Outline

- Supervised learning: Classification
- Medical diagnosis using classification
- Naive Bayes classifiers: model fitting and prediction
- Model validation: assessing prediction accuracy
- Mitigate overfitting with Bayesian inference
- Feature selection

Feature selection

- Consider big datasets with large number of features d
- To reduce the cost of our algorithm and complexity of our model, we would like to remove features that are not so relevant for the classification task
- This is known as **feature selection**
- We will consider an approach called **variable ranking**:
    1. evaluate the relevance of each feature separately;
    2. select top K features
- We then select the value of K that minimizes the **misclassification rate**

Mutual information

- To measure relevance, we use the **mutual information** between feature X_i and the class label Y

$$I(X_i, Y) = \sum_{x_i} \sum_{y} \log \left\{ \frac{p(x_i, y)}{p(x_i)p(y)} \right\} p(x_i, y)$$

- $I(X_i, Y) \geq 0$ and $I(X_i, Y) = 0$ if X_i and Y are independent

- The above probabilities can be computed after fitting the model