SA22011021 Yi-Xiang Hu
School of Computer Science and Technology
University of Science and Technology of China

Exercise Sheet 1 for
Applied Mathematic for Computer Science and Technology
Autumn 2022

Due 10 Nov 2022 at 23:59

Exercise 1

$$f(x) = \begin{cases} kx, 0 \leq x < 3 \\ 2 - \dfrac{x}{2}, 3 \leq x < 4 \\ 0, others \end{cases}$$

$$\int_{-\infty}^{\infty} f(x) = \int_{0}^{3} kxdx + \int_{3}^{4} 2 - \frac{x}{2} = \frac{1}{4} + \frac{9k}{2}dx$$

$$k = \frac{1}{6}$$

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx = \frac{7}{3}$$

$$E(X^2)\int_{-\infty}^{\infty} x^2 f(x)dx = \frac{37}{6}$$

$$Var(X) = E(X^2) - E(X)^2 = \frac{37}{6} - \left(\frac{7}{3}\right)^2 = \frac{13}{18}$$

Exercise 2

$$SE_X = \frac{\sigma}{\sqrt{n}} = \frac{\sqrt{2.65}}{\sqrt{18}} = 0.38$$

95% 的置信区间的显著性水平 $\alpha = 0.05, Z_{\alpha/2} = 1.96$
所以 95% 的置信区间为

$$24.9 - 1.96 \times SE_X \leq u \leq 24.9 + 1.96 \times SE_X$$

即 $[24.16, 25.64]$
99% 的置信区间的显著性水平 $\alpha = 0.01, Z_{\alpha/2} = 2.58$
所以 95% 的置信区间为

$$24.9 - 2.58 \times SE_X \leq u \leq 24.9 + 2.58 \times SE_X$$

即 $[23.92, 25.88]$

Exercise 3

$$SE_X = \frac{S}{\sqrt{n-1}} = \frac{10}{\sqrt{19}} = 2.29$$

$$t_{0.05/2(19)} = 2.093 \; t_{0.01/2(19)} = 2.861$$

所以 *95%* 的置信区间为

$$155 - 2.093 \times 2.29 \leq u \leq 155 + 2.093 \times 2.29$$

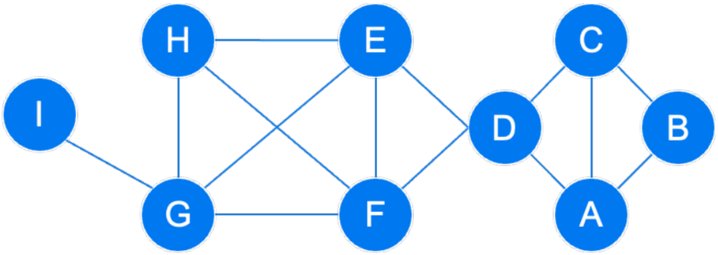
即 $[150.21, 159.79]$

所以 *99%* 的置信区间为

$$155 - 2.861 \times 2.29 \leq u \leq 155 + 2.861 \times 2.29$$

即 $[148.45, 161.55]$

Exercise 4



$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

$$g(A) = g(C) = \sum_{s \neq A} \frac{\sigma_{sB(A)}}{\sigma_{sB}} = 6 \times \frac{1}{2} + 0 = 3$$

$$g(B) = 0$$

$$g(D) = \sum_{s \in \{A,B,C\}, t \in \{E,F,G,H,I\}} \frac{\sigma_{st(D)}}{\sigma_{st}} = |\{A, B, C\}| \times |\{E, F, G, H, I\}| = 3 \times 5 = 15$$

$$g(E) = g(F) = \sum_{s \in \{A,B,C\}, t \in \{G,H,I\}} \frac{\sigma_{st(E)}}{\sigma_{st}} = \frac{1}{2} \times |\{A, B, C, D\}| \times |\{G, H, I\}| = \frac{1}{2} \times 3 \times 5 = 6$$

$$=$$

$$g(G) = \sum_{s \in \{A,B,C,D,E,F,H\}} \frac{\sigma_{sI(G)}}{\sigma_{sI}} = |\{A, B, C, D, E, F, H\}| = 7$$

$$g(H) = 0$$

$$g(I) = 0$$

Exercise 5 *Naive Bayes Classifier*

源代码:

```
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, ComplementNB
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import f1_score, recall_score, precision_score
import pandas as pd
# load data
train_file = 'D:\\AMCS_2022\\data\\Bayesian_Dataset_train.csv'  # training set
df = pd.read_csv(train_file, header=None)
X_train = df.iloc[:, :-1]
y_train = df.iloc[:, -1]

test_file = 'D:\\AMCS_2022\\data\\Bayesian_Dataset_test.csv'  # test set
df = pd.read_csv(test_file, header=None)
X_test = df.iloc[:, :-1]
X_test_before_transform = X_test
y_test = df.iloc[:, -1]

X = pd.concat([X_test, X_train])

# data preprocessing

OrdEnc = OrdinalEncoder()
X = OrdEnc.fit_transform(X)
X = pd.DataFrame(X)
X_test = X.iloc[:len(X_test), :]
X_train = X.iloc[len(X_test):, :]

# naive bayes
# select Naive Bayes classifier
# and set parameters
# nb = BernoulliNB()  # Naive Bayes classifier for multivariate Bernoulli models.
# nb = GaussianNB(var_smoothing=1e-9)  # Gaussian Naive Bayes
nb = CategoricalNB(alpha=1.0,fit_prior=True)  # Naive Bayes classifier for categorical features.
# nb = ComplementNB()  # Complement Naive Bayes classifier
# nb = MultinomialNB()  # Naive Bayes classifier for multinomial models
y_pred = nb.fit(X_train, y_train).predict(X_test)

# print result statistics
print("positive label : ' >50K'")
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test != y_pred).sum()))
```

```
print("Accuracy = %.4f" % (1-(y_test != y_pred).sum()/X_test.shape[0]))
print("Precision = %.4f" % (precision_score(y_test, y_pred, pos_label=' >50K')))
print("Recall = %.4f" % (recall_score(y_test, y_pred, pos_label=' >50K')))
print("F1_score = %.4f" % (f1_score(y_test, y_pred, pos_label=' >50K')))

# save prediction
y_pred = pd.Series(y_pred)
result = pd.concat([X_test_before_transform, y_test, y_pred], axis=1)
result.to_csv("D:\\AMCS_2022\\data\\Bayesian_Dataset_pred.csv", header=0)
```

结果:
计算 Precision，Recall，F1 score 时，将”>50K” 作为正标签。

	BernoulliNB	CategoricalNB	ComplementNB	GaussianNB	MultinomialNB
Accuracy	0.7301	0.7916	0.5916	0.7608	0.6172
Precision	0.4774	0.5630	0.3252	0.5207	0.3345
Recall	0.7206	0.7821	0.5735	0.6725	0.5201
F1 score	0.5743	0.6547	0.4151	0.5869	0.4071

表 1: 不同朴素贝叶斯分类器在默认参数下的性能

根据表 1结果可知，Categorical Naive Bayes Classifier 和 Gaussian Naive Bayes Classifier 的性能较好，选取这两个分类器设置不同参数作对比。
Categorical Naive Bayes Classifier 的可选参数有:

(a) alpha float, default=1.0 Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

(b) fit_prior bool, default=True Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

(c) class_prior array-like of shape (n_classes,), default=None Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.

(d) min_categories int or array-like of shape (n_features,), default=None Minimum number of categories per feature.

  - integer: Sets the minimum number of categories per feature to n_categories for each features.
  - array-like: shape (n_features,) where n_categories[i] holds the minimum number of categories for the ith column of the input.
  - None (default): Determines the number of categories automatically from the training data.

对比实验选取 alpha，fit_prior 两个参数。

alpha	1.0(default)	1.0	1.0e-10	1.0e-10	2.0	2.0
fit_prior	True(default)	False	True	False	True	False
Accuracy	0.7916	0.7490	0.7372	0.7162	0.7946	0.7514
Precision	0.5630	0.5020	0.4851	0.4585	0.5686	0.5048
Recall	0.7821	0.8516	0.6537	0.6791	0.7754	0.8489
F1 score	0.6547	0.6316	0.5569	0.5474	0.6561	0.6331

表 2: 分类朴素贝叶斯分类器在不同参数设置下的性能

Categorical Naive Bayes Classifier 的可选参数有:

(a) priors array-like of shape (n_classes,) Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.

(b) var_smoothing float, default=1e-9 Portion of the largest variance of all features that is added to variances for calculation stability.

对比实验选取参数 var_smoothing。

var_smoothing	1e-9(default)	1e-11	1e-10	1e-8	1e-7
Accuracy	0.7608	0.7571	0.7574	0.7693	0.7618
Precision	0.5207	0.5147	0.5152	0.5377	0.5433
Recall	0.6725	0.6791	0.6791	0.6190	0.3610
F1 score	0.5869	0.5856	0.5859	0.5755	0.4337

表 3: 高斯朴素贝叶斯分类器在不同参数设置下的性能

Exercise 6 *Gaussian Mixture Model*

源代码:

```
from sklearn import datasets
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt
# load data
X, y = datasets.load_iris(return_X_y=True)

# Gaussian Mixture Model
# set covariance_type as spherical, diagonal, tied or full
gm = GaussianMixture(n_components=3, covariance_type='full',
                     random_state=0).fit(X, y)

# print mean, variance and weight
print("mean : ", gm.means_)
print("variance : ", gm.covariances_)
print("weight : ", gm.weights_)

# predict
labels = gm.predict(X)

# plot
fig = plt.figure(figsize=(12, 6))
ax = fig.add_subplot(121, projection='3d')
for i in range(len(y)):
    if y[i] == 0:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='r')
    elif y[i] == 1:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='blue')
    elif y[i] == 2:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='g')
ax.set_title("truth")

ax = fig.add_subplot(122, projection='3d')
for i in range(len(labels)):
    if labels[i] == 0:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='yellow')
    elif labels[i] == 1:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='grey')
    elif labels[i] == 2:
        ax.scatter(X[i][0], X[i][1], X[i][2], c='brown')
ax.set_title("predict")
plt.savefig('D:\\AMCS_2022\\result\\result.png')
plt.show()
```

covariance structures 为 full 时的结果：

- mean:

$$\begin{bmatrix} 5.006 & 3.428 & 1.462 & 0.246 \\ 6.54639415 & 2.94946365 & 5.48364578 & 1.98726565 \\ 5.9170732 & 2.77804839 & 4.20540364 & 1.29848217 \end{bmatrix}$$

- variance:

$$\begin{bmatrix} 0.121765 & 0.097232 & 0.016028 & 0.010124 \\ 0.097232 & 0.140817 & 0.011464 & 0.009112 \\ 0.016028 & 0.011464 & 0.029557 & 0.005948 \\ 0.010124 & 0.009112 & 0.005948 & 0.010885 \end{bmatrix}$$

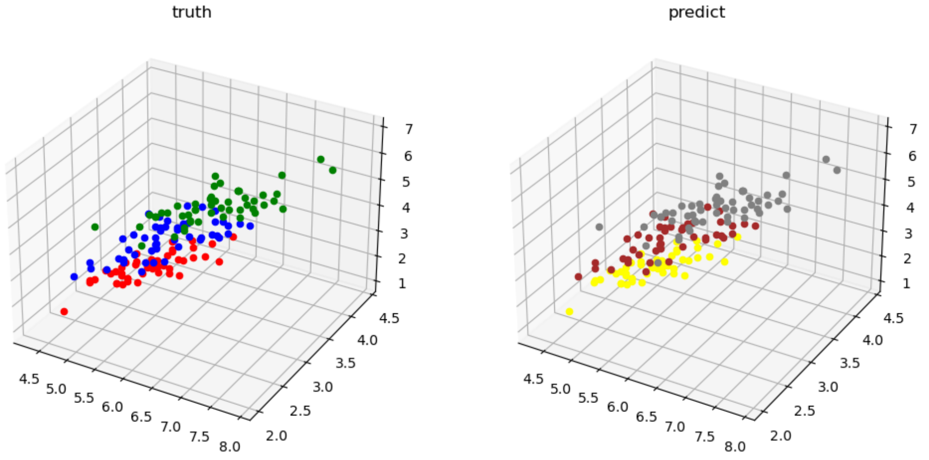
$$\begin{bmatrix} 0.38744093 & 0.09223276 & 0.30244302 & 0.06087397 \\ 0.09223276 & 0.11040914 & 0.08385112 & 0.05574334 \\ 0.30244302 & 0.08385112 & 0.32589574 & 0.07276776 \\ 0.06087397 & 0.05574334 & 0.07276776 & 0.08484505 \end{bmatrix}$$

$$\begin{bmatrix} 0.2755171 & 0.09662295 & 0.18547072 & 0.05478901 \\ 0.09662295 & 0.09255152 & 0.09103431 & 0.04299899 \\ 0.18547072 & 0.09103431 & 0.20235849 & 0.06171383 \\ 0.05478901 & 0.04299899 & 0.06171383 & 0.03233775 \end{bmatrix}$$

- weight:

$$\begin{bmatrix} 0.33333333 & 0.36539575 & 0.30127092 \end{bmatrix}$$

- distribution:



covariance structures 为 spherical 时的结果：

- mean:

$$\begin{bmatrix} 5.006 & 3.428 & 1.462 & 0.246 \\ 6.84191531 & 3.0717748 & 5.72292746 & 2.07049253 \\ 5.90165943 & 2.74786641 & 4.39838776 & 1.43087513 \end{bmatrix}$$

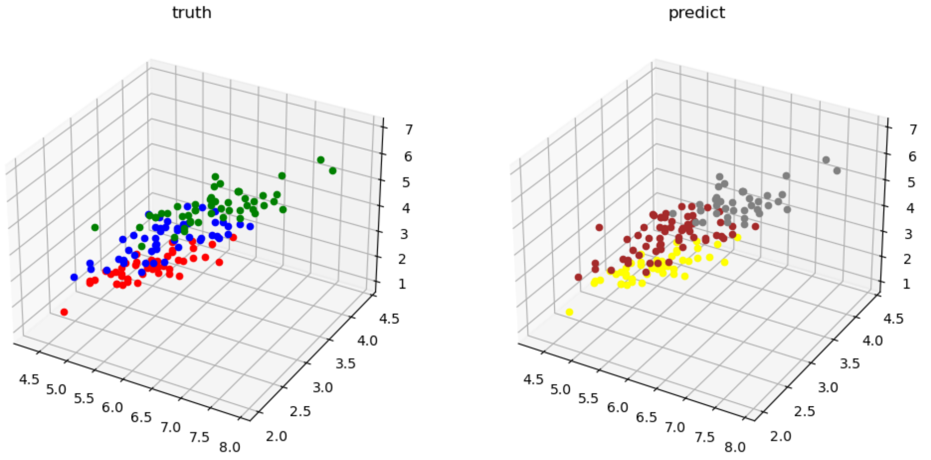
- variance:

$$\begin{bmatrix} 0.075756 & 0.1644286 & 0.16244304 \end{bmatrix}$$

- weight:

$$\begin{bmatrix} 0.33333333 & 0.25549114 & 0.41117552 \end{bmatrix}$$

- distribution:



covariance structures 为 diagonal 时的结果：

- mean:

$$\begin{bmatrix} 5.006 & 3.428 & 1.462 & 0.246 \\ 6.8060823 & 3.07023103 & 5.71889409 & 2.10305305 \\ 5.92570673 & 2.74947486 & 4.40355614 & 1.41204165 \end{bmatrix}$$

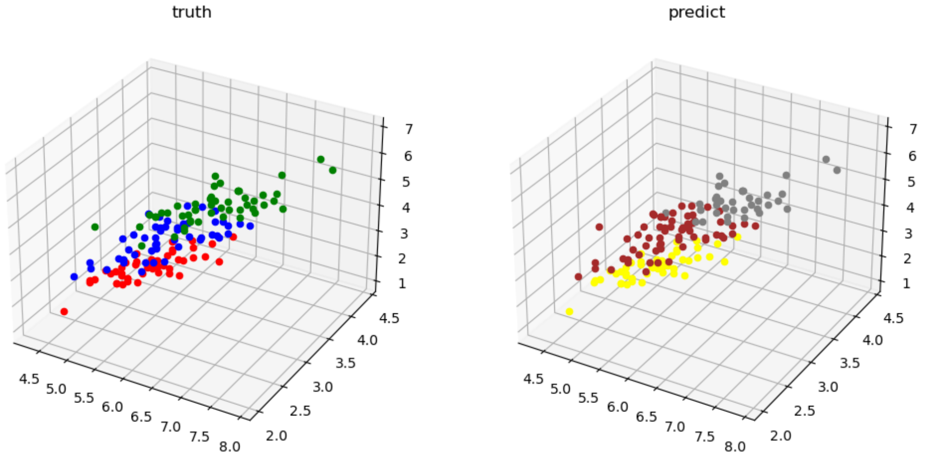
- variance:

$$\begin{bmatrix} 0.121765 & 0.140817 & 0.029557 & 0.010885 \\ 0.28520059 & 0.08200975 & 0.25128562 & 0.06109484 \\ 0.23145596 & 0.08738014 & 0.27563102 & 0.0688713 \end{bmatrix}$$

- weight:

$$\begin{bmatrix} 0.33333333 & 0.25465894 & 0.41200772 \end{bmatrix}$$

- distribution:



covariance structures 为 tied 时的结果：

- mean:

$$\begin{bmatrix} 5.006 & 3.428 & 1.462 & 0.246 \\ 6.59293638 & 2.99716202 & 5.55284949 & 2.05143145 \\ 5.95728298 & 2.75675429 & 4.31039897 & 1.33031316 \end{bmatrix}$$

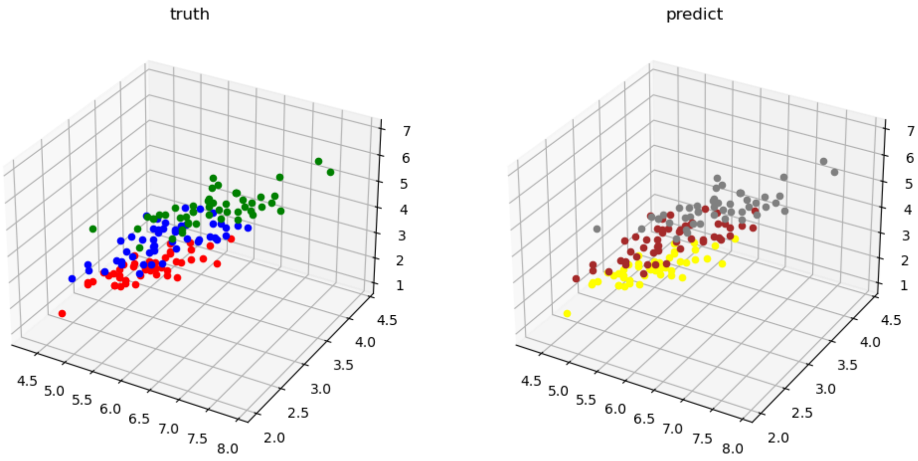
- variance:

$$\begin{bmatrix} 0.2633317 & 0.08760867 & 0.1731573 & 0.0374331 \\ 0.08760867 & 0.11040074 & 0.04836891 & 0.02701142 \\ 0.1731573 & 0.04836891 & 0.20285285 & 0.04347376 \\ 0.0374331 & 0.02701142 & 0.04347376 & 0.03619053 \end{bmatrix}$$

- weight:

$$\begin{bmatrix} 0.33333333 & 0.31958404 & 0.34708263 \end{bmatrix}$$

- distribution:


truth
predict