รายงาน

เรื่อง การทดลอง Minimum Risk Bayes Decision Theoretic Classification

จัดทำโดย

นายชยดนย์ ลัพบุตร

รหัสนักศึกษา 660631095

เสนอ

อาจารย์ รศ.ดร.ศันสนีย์ เอื้อพันธ์วิริยะกุล

รายงานนี้เป็นส่วนหนึ่งของรายวิชา CPE 262754 Advanced Pattern Recognition

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

ภาคเรียนที่ 1 ปีการศึกษา 2566

มหาวิทยาลัยเชียงใหม่

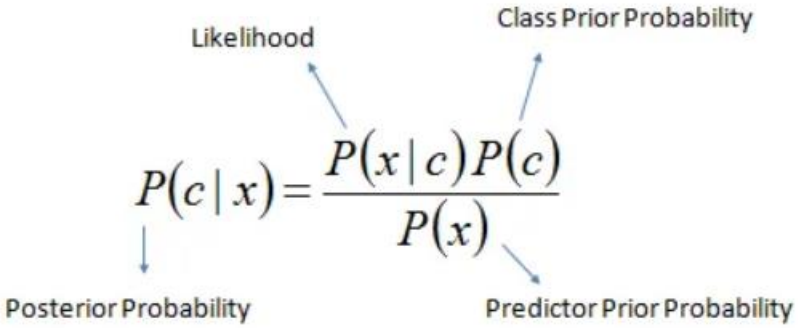
สารบัญ

	หน้า
1. รายละเอียดของทฤษฎีหรือวิธีการต่าง ๆ ที่ใช้	1
2. Algorithm	4
3. ผลการทดลอง	5
4. การวิเคราะห์ผลการทดลอง	6
5. ภาคผนวก	7
6. อ้างอิง	11

รายละเอียดของทฤษฎีหรือวิธีการ

1.1 Bayes Classification

เป็นการประยุกต์ใช้ทฤษฎีความน่าจะเป็นของ Bayes มาทำการจัดกลุ่มของข้อมูล ซึ่งสามารถเขียนเป็นสมการได้ ดังนี้



จากสมการมีตัวแปรทั้งหมด 3 ตัวได้แก่

- c คือ Class
- x คือ Attribute
- P คือ Probability

P(c|x) Posterior probability คือ ความน่าจะเป็นที่ข้อมูลที่มี attribute เป็น x จะอยู่ในคลาส C

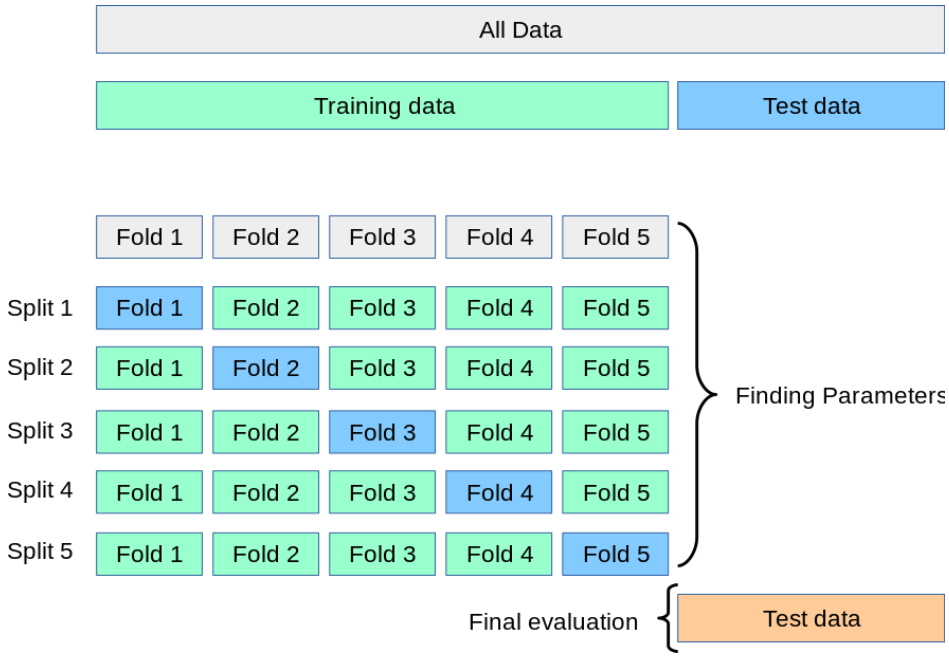
P(x|c) Likelihood คือ ความน่าจะเป็นที่ข้อมูลที่มีคลาส C และมี attribute เป็น x

P(c) Prior probability คือ ความน่าจะเป็นของคลาส c

P(x) Predictor Prior probability คือ ความน่าจะเป็นของคลาส x

การนำมาใช้แบ่งกลุ่ม (Classification) จะต้องมีการเปรียบเทียบกันระหว่างกลุ่มอย่างน้อย 2 กลุ่ม โดยใช้ค่าความน่าจะเป็นของ *x* จากโมเดลหรือฟังก์ชันไหนมีค่ามากที่สุด ก็สามารถอนุมานได้ว่า *x* น่าจะอยู่ใน class นั้น

1.2 K-Fold Cross Validation



K-Fold Validation หรือ k-fold cv เป็นเทคนิคที่ใช้ในการสร้างและทดสอบโมเดล Machine Learning และเป็นหนึ่งในเทคนิคของการทำ Resampling โดยขั้นตอนของการทำงานของเทคนิคนี้คือ การแบ่งข้อมูลออกเป็น k ส่วนที่เท่า ๆ กันเพื่อใช้สร้างและทดสอบโมเดล แล้วคำนวณค่าเฉลี่ย Accuracy หรือ Error ก่อนนำไปทดสอบด้วย test set ซึ่งโดยทั่วไปแล้วจำนวน k ที่นิยมใช้กันมีอยู่สองค่า คือ k = 5, k = 10

1.3 Confusion Matrix

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Confusion Matrix คือ matrix ที่ใช้ในการประเมินผลลัพธ์ของการทำนายจากโมเดลที่สร้างขึ้น มีค่าดังนี้

True Positive(TP) = สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง ในกรณี ทำนายว่าจริงและสิ่งที่เกิดขึ้น ก็คือ จริง

True Negative(TN)= สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณีทำนายว่า ไม่จริง และสิ่งที่เกิดขึ้นก็คือ ไม่จริง

False Positive(FP) = สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้นคือ ไม่จริง

False Negative(FN) = สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้นคือ จริง

โดย TP,TN,FP,FN ในตารางจะแทนด้วยค่าความถี่

ซึ่งทั้ง 4 ค่านี้จะถูกนำมาคำนวณค่า Accuracy ตามสูตรด้านล่างนี้

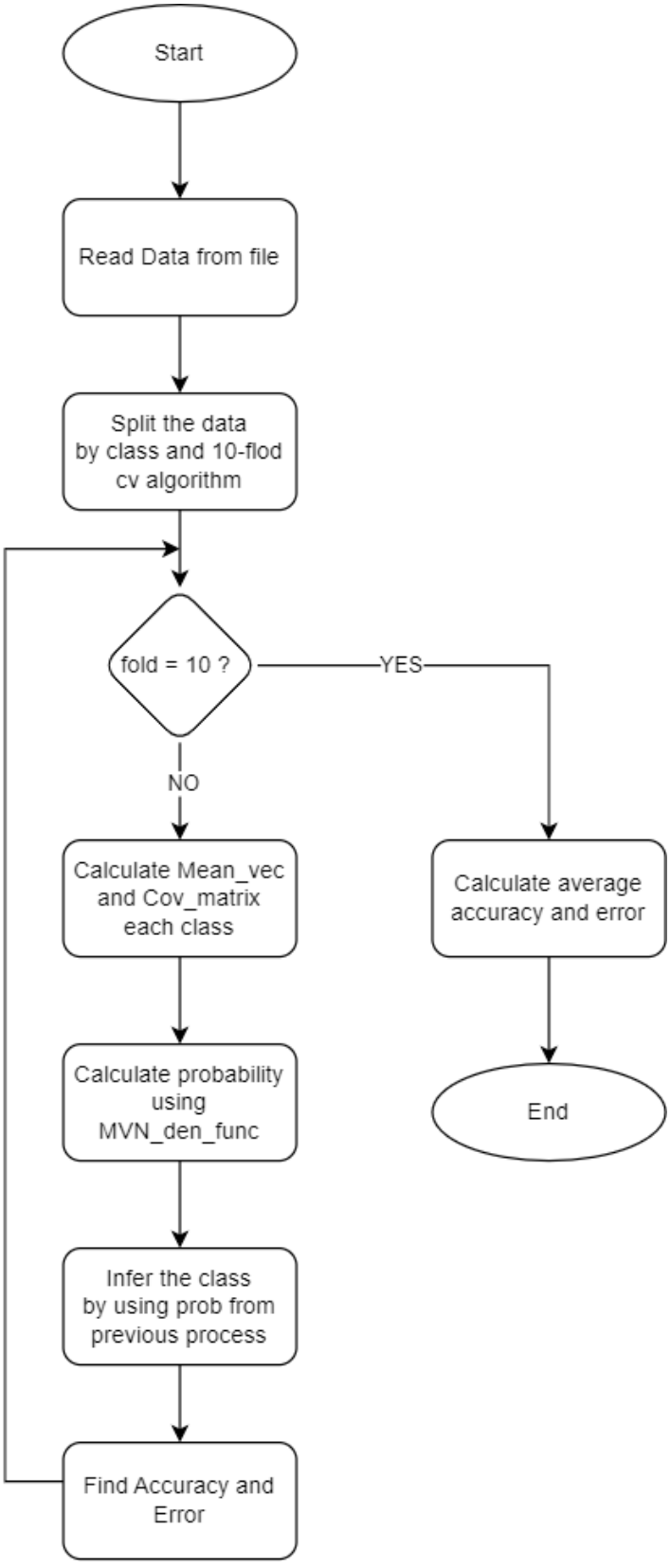
$$Overall\ Accuracy = \frac{TP + TN}{N}$$

และสามารถหาค่า ERROR ได้จาก

$$Error = 1 - Overall\ Accuracy$$

Algorithm

Overview process of program by flowchart



ผลการทดลอง

ผลการทดลองที่ใช้ 4 classes และตัวอย่าง Fold

```
------------------------------ folds 10 ------------------------------
Class 1: - Mean Vector
[6.3504, 2.9217, 4.2156, 1.2948]
Class 1: - Covariance Matrix
[0.30416984, -0.01924768, 0.02609876, -0.01157192]
[-0.01924768, 0.11199611, -0.00755052, 0.00573884]
[0.02609876, -0.00755052, 0.15171664, 0.01483312]
[-0.01157192, 0.00573884, 0.01483312, 0.03876896]

Class 2: - Mean Vector
[6.593625, 2.908375, 5.61975, 2.07675]
Class 2: - Covariance Matrix
[0.3939581094, 0.0105696406, -0.0126328437, -0.0012357187]
[0.0105696406, 0.0988661094, 0.0190945937, -0.0028577812]
[-0.0126328437, 0.0190945937, 0.1634424375, 0.0239429375]
[-0.0012357187, -0.0028577812, 0.0239429375, 0.0608419375]

worng inference :  [6.76, 2.38, 4.79, 2.06]
worng inference :  [6.73, 2.86, 5.0, 1.83]
worng inference :  [6.84, 2.93, 5.12, 1.48]
worng inference :  [7.04, 3.57, 5.03, 1.26]
Accuracy =  80.0
Confusion Matrix:
[0, 0]
[4, 16]
--------------------------------------------------------------------------
```

```
Average accuracy = 91.0
Average error = 9.0
```

ผลการทดลองที่ใช้ 2 classes และตัวอย่าง Fold

```
------------------------------ folds 10 ------------------------------
Class 1: - Mean Vector
[6.3504, 2.9217]
Class 1: - Covariance Matrix
[0.30416984, -0.01924768]
[-0.01924768, 0.11199611]

Class 2: - Mean Vector
[6.593625, 2.908375]
Class 2: - Covariance Matrix
[0.3939581094, 0.0105696406]
[0.0105696406, 0.0988661094]

worng inference :  [5.33, 2.98]
worng inference :  [6.84, 2.44]
worng inference :  [6.76, 2.38]
worng inference :  [6.1, 2.72]
worng inference :  [6.73, 2.86]
worng inference :  [6.25, 3.08]
worng inference :  [6.49, 2.78]
worng inference :  [5.76, 2.79]
worng inference :  [6.08, 2.85]
worng inference :  [6.55, 3.14]
worng inference :  [6.64, 2.96]
Accuracy =  45.0
Confusion Matrix:
[0, 0]
[11, 9]
```

```
Average accuracy = 56.0
Average error = 44.0
```

การวิเคราะห์ผลการทดลอง

ผลการทำนายหลังจากที่ตัด features ที่ 3 และ 4 แล้วได้ accuracy ที่ลดลงคิดว่าน่าจะเป็นผลมาจากข้อมูลที่น้อยลงอาจไม่เพียงพอในการให้ข้อมูลที่เป็นประโยชน์ในการเรียนรู้และความสามารถของโมเดล ทำให้โมเดลไม่สามารถทำนายอย่างแม่นยำ จากค่าความแม่นยำที่ลดลงจึงได้นำข้อมูลไปหา Correlation Coefficient และ Variance ในแต่ละ feature ได้ผลตามรูปด้านล่าง

	f1	*f2*	*f3*	*f4*
f1	1			
f2	-0.04371	1		
f3	0.19895	-0.01894	1	
f4	0.14885	-0.01831	0.78533	1

variance	*f1*	*f2*	*f3*	*f4*
	0.35056	0.10616	0.66362	0.20262

Feature 3 และ Feature 4 มีค่า Positive correlation ที่สูงจึงได้ทดสอบโดย Feature 4 ออกผลลัพธ์ที่ได้คือ ค่าความแม่นยำเฉลี่ยเพิ่มขึ้นมาเป็น 95% แต่เมื่อทดลองนำ Feature 3 ออก ค่าความแม่นยำเฉลี่ยที่ได้คือ 73%

ภาคผนวก

- Code

```
#########################################################################
# Program: Minimum Risk Decision Theoretic Classifier
# Author: Chayadon Lappabud
# Date: July 26, 2023
# Description: very hard. T T
#########################################################################
import math  as m

transpost = lambda x: list(zip(*x))
print_matrix = lambda x: list(map(print, x))

# Create function reading dataset.
def read_data(file_path:str)->list:
    data = []
    with open(file_path, "r") as file:
        lines = file.readlines()
        lines = lines[1:]
        for line in lines:
            values = line.strip().split()
            data.append([float(val) for val in values])
    return data


# Calculate mean vector.
def cal_mean_vector(data:list)->list:
    num_of_features = len(data[0])
    num_of_samples = len(data)
    mean_vector = []
    for i in range(num_of_features):
        sum = 0
        for j in range(num_of_samples):
            sum += data[j][i]
        mean_vector.append(round(sum / num_of_samples, 10))
    return mean_vector


# Define function cal_covariance_matrix.
def cal_covariance_matrix(mean_vec:list, data_class:list)->list:
    num_of_sample = len(data_class)
    cov = [[0 for y in data_class[0]] for _ in data_class]
    i = 0
    for sample in data_class:
        j = 0
        for cols in sample:
            cov[i][j] = round(cols - mean_vec[j], 10)
            j += 1
        i += 1
    inv_cov = transpost(cov)
    covariance = [[round(sum(a * b for a, b in zip(inv_cova, cova))/num_of_sample,10)
                                    for cova in zip(*cov)]
                                        for inv_cova in inv_cov]
    return covariance
```

```
# Define function train_test_split to split the data and labels.
def train_test_split(dataset:list)->list:
    data = [d[:-1] for d in dataset]
    labels = [int(label[-1]) for label in dataset]
    return data, labels


# Define function split_by_class.
def split_by_class(dataset:list, labels:list, classes:int)->list:
        return [dataset[i] for i in range(len(dataset)) if labels[i] == classes]


# Define this function to calculate multivariate normal prob density function.
def multivariate_normal_prob_density_function(x:list, mean:list, covariance_matrix:list)-
>float:
    num_features = len(x)
    det = 1.0
    inverse_covariance_matrix = []

    for i in range(num_features):
        det *= covariance_matrix[i][i]
        inverse_row = [-c / covariance_matrix[i][i] for c in covariance_matrix[i]]
        inverse_row[i] = -inverse_row[i]
        inverse_covariance_matrix.append(inverse_row)

    prefactor = m.pow((2 * m.pi), -num_features/2) * det**(-1/2)
    exponent = 0.0

    for i in range(num_features):
        for j in range(num_features):
            exponent += (x[i] - mean[i]) * inverse_covariance_matrix[i][j] * (x[j] -
mean[j])
    return prefactor * m.exp(-0.5 * exponent)


# Predict the class of given data.
def inference(data:list, mean1:list, covariance_matrices1:list, mean2:list,
covariance_matrices2:list)->int:
    predicted_labels = []
    means = [mean1, mean2]
    covariance_mat = {1:covariance_matrices1, 2:covariance_matrices2}

    for sample in data:
        max_posterior_prob = 0
        predicted_label = None
        for mean,class_label in zip(means, [1,2]):
            posterior_prob = multivariate_normal_prob_density_function(sample,
                                                                       mean,
                                                                covariance_mat[class_label])
            if posterior_prob > max_posterior_prob:
                max_posterior_prob = posterior_prob
                predicted_label = class_label
        predicted_labels.append(predicted_label)

    return predicted_labels
```

```
# Define this function to train model.
def model_fit(i:int, test_data:list, test_label:list, *dataclass:list)->int:
    print(f"------------------------------ folds {i+1} ------------------------------")
    for j in range(len(dataclass)):
        print(f"Class {j+1}: - Mean Vector")
        vector_mean = cal_mean_vector(dataclass[j])
        print(vector_mean)

        print(f"Class {j+1}: - Covariance Matrix")
        cov_mat = cal_covariance_matrix(vector_mean, dataclass[j])
        print_matrix(cov_mat)
        print()

    vector_mean = cal_mean_vector(dataclass[0])
    vector_mean2 = cal_mean_vector(dataclass[1])

    cov_mat = cal_covariance_matrix(vector_mean, dataclass[0])
    cov_mat2 = cal_covariance_matrix(vector_mean2, dataclass[1])

    predicted = inference(test_data, vector_mean, cov_mat, vector_mean2, cov_mat2)

    correct = 0
    for i in range(len(test_data)):
        if test_label[i] == predicted[i]:
            correct += 1
        else:
            print("worng inference : ", test_data[i])

    print("Accuracy = ", (correct/len(test_label))*100)
    print("Confusion Matrix:")
    conf_matrix = confusion_matrix(test_label, predicted)
    print_matrix(conf_matrix)
    print(f"-------------------------------------------------------------------------")
    return correct

# Calculate confusion matrix and return it
def confusion_matrix(actual_labels:list, predicted_labels:list)->list:
    confusion_matrix = [[0,0],[0,0]]
    for actual_label, predicted_label in zip(actual_labels, predicted_labels):
        actual_idx = int(actual_label) - 1
        predicted_idx = int(predicted_label) - 1
        confusion_matrix[actual_idx][predicted_idx] += 1
    return confusion_matrix


# Define function to do k-folds cross_validation.
def cross_validation(dataset:dict, folds = 10):
    fold_size = len(dataset) // folds
    data, labels = train_test_split(dataset)
    accuracy = 0
    for i in range(folds):
        idx_start = i * fold_size
        idx_end = fold_size * (i+1)

        X_train = data[:idx_start] + data[idx_end:]
        Y_train = labels[:idx_start] + labels[idx_end:]
```

```
        x_test = data[idx_start:idx_end]
        y_test = labels[idx_start:idx_end]

        class_1 = split_by_class(X_train, Y_train, 1)
        class_2 = split_by_class(X_train, Y_train, 2)

        accuracy += model_fit(i, x_test, y_test, class_1, class_2)

    print(f'Average accuracy = {round(accuracy/len(dataset)*100,4)}')
    print(f'Average error = {round((1-accuracy/len(dataset))*100,4)}')


# MAIN CODE.
if __name__ == "__main__":
    dataset = read_data('TWOCLASS.txt')
    cross_validation(dataset)
```

อ้างอิง

Kong Ruksiam. (2563). สรุป Machine Learning(EP.6)-การจัดหมวดหมู่ด้วย Naive Bayes. สืบค้นเมื่อ 2/8/2566, จาก https://kongruksiam.medium.com/สรุป-machine-learning-ep-5-การจัดหมวดหมู่ด้วย-naive-bayes-eb9ce0e1b010

Peachapong Poolpol. (2564). Naïve Bayes Classification. สืบค้นเมื่อ 2/8/2566, จาก https://peachapong-poolpol.medium.com/na%C3%AFve-bayes-classification-cb6cf905505d

Kasidis Satangmongkol. (2562). อธิบาย K-Fold Cross Validation พร้อมโค้ดตัวอย่างใน R. สืบค้นเมื่อ 2/8/2566, จาก https://datarockie.com/blog/k-fold-cross-validation/

Kasidis Satangmongkol. (2566). Confusion Matrix คืออะไร พร้อมวิธีคำนวณค่าสถิติง่ายๆ. สืบค้นเมื่อ 2/8/2566,จากttps://datarockie.com/blog/confusion-matrix-explained/#:~:text=Confusion%20matrix%20คือตาราง%20cross,cross%20กันตามชื่อเลย