รายงาน

เรื่อง k-NN

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

รายละเอียดของทฤษฎีหรือวิธีการ

1.1 k-NN Algorithm

k-Nearest Neighbors (k-NN) เป็นอัลกอริทึมที่ใช้ในการจำแนกประเภทข้อมูล (classification) หรือการทำนายค่าตัวเลข (regression) โดยการใช้ข้อมูลตัวอย่างในการตัดสินใจว่าข้อมูลป้อนเข้า (query data) จะอยู่ในกลุ่มไหนหรือมีค่าเท่าไหร่ อัลกอริทึมนี้พึงระวังถึงเพื่อนบ้าน (neighbors) ที่ใกล้ที่สุดกับข้อมูลป้อนเข้า โดยใช้ระยะห่าง (distance) ระหว่างข้อมูลเหล่านี้เป็นตัวกำหนด ซึ่งระยะห่างที่ส่วนมากนิยมใช้คือ Euclidean distance และสมการของ k-NN สามารถเขียนแทนด้วยการอธิบายขั้นตอนหลัก ๆ ดังนี้:

1.1.1 การคำนวณระยะห่าง (Distance Calculation):

ให้ $\mathrm{d}(x, x')$ เป็นระยะห่างระหว่างข้อมูลป้อนเข้า x กับข้อมูลตัวอย่าง x' โดยใช้ฟังก์ชันระยะห่าง (distance function) เช่น Euclidean distance:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2}$$

โดยที่ x_i และ x_i' คือคุณสมบัติ (features) ของข้อมูล x_i และ x_i' ในมิติที่ i และ n คือจำนวนของคุณสมบัติทั้งหมด

1.1.2 การเลือกเพื่อนบ้าน (Neighbor Selection):

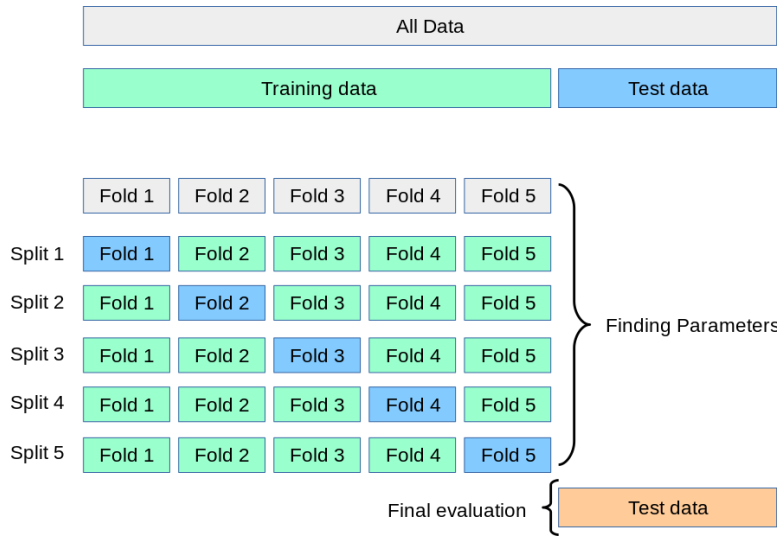
หลังจากคำนวณระยะห่างระหว่างข้อมูลป้อนเข้า x กับข้อมูลตัวอย่างทั้งหมดในชุดการฝึก (training dataset) ให้เรียงลำดับข้อมูลตัวอย่างตามระยะห่างจากน้อยไปมากและเลือกเพื่อนบ้าน k คนที่ใกล้ที่สุดกับ x เพื่อใช้ในขั้นตอนถัดไป.

1.1.3 การคำนวณผลลัพธ์ (Prediction):

สำหรับงานการจำแนกประเภท (classification), ให้นับจำนวนของแต่ละประเภทของเพื่อนบ้าน k คนและเลือกประเภทที่มีจำนวนมากที่สุดเป็นผลลัพธ์ที่ทำนาย (predicted class). สำหรับงานการทำนายค่าตัวเลข (regression), ให้หาค่าเฉลี่ยของค่าตัวเลขของเพื่อนบ้าน k คนและใช้ค่านี้เป็นผลลัพธ์ที่ทำนาย (predicted value).

สมการของ k-NN นั้นไม่มีตรง ๆ แต่ขั้นตอนที่ได้กล่าวถึงสามารถอธิบายความหมายและหลักการทำงานของอัลกอริทึมได้โดยทั่วไป การใช้ k-NN จะต้องระบุรายละเอียดเพิ่มเติมเกี่ยวกับข้อมูลเชิงพื้นฐานและการคำนวณที่ใช้ในแต่ละกระบวนการและโครงสร้างของตัวอย่างข้อมูลและชุดการฝึก.

1.2 K-Fold Cross Validation



Cross-validation เป็นเทคนิคทางสถิติที่ใช้ในการประเมินประสิทธิภาพของแบบจำลอง (model) โดยแบ่งข้อมูลออกเป็นชุดการฝึก (training) และชุดทดสอบ (testing) โดยที่ชุดการฝึกถูกใช้ในการฝึกแบบจำลองและชุดทดสอบถูกใช้ในการทดสอบแบบจำลอง นิยมใช้เมื่อไม่มีข้อมูลทดสอบแยกไว้สำหรับการประเมินแบบจำลองในข้อมูลจริง หรือเพื่อประหยัดข้อมูลในกรณีที่ข้อมูลมีจำนวนมาก หรือเพื่อป้องกันการเกิด overfitting ในการฝึกแบบจำลอง

ทฤษฎีของ Cross-validation จะอธิบายการทำงานของเทคนิคนี้ โดยปกติแล้วมีหลายวิธีการในการทำ Cross-validation แต่ต่อไปนี้เราจะพูดถึงวิธีการ "k-fold Cross-validation" ที่เป็นอีกหนึ่งรูปแบบที่นิยมใช้บ่อย โดยในวิธีนี้ข้อมูลจะถูกแบ่งเป็น k ชุดที่เท่าๆ กัน (เรียกว่า "folds") และในแต่ละรอบการทดสอบ จะใช้ 1 ชุดเป็นชุดทดสอบ และ k-1 ชุดเป็นชุดการฝึก จากนั้นคำนวณประสิทธิภาพของแบบจำลอง และทำกระบวนการนี้อีกรอบเป็นจำนวน k ครั้ง โดยแบบจำลองที่ทดสอบในแต่ละรอบอาจเป็นแบบจำลองที่แตกต่างกันไป จากนั้นคำนวณค่าเฉลี่ยหรือค่าสถิติที่เกี่ยวข้องจากผลการทดสอบในทุกๆ รอบ เพื่อประเมินความสามารถของแบบจำลองทั้งหมด

สมการที่อธิบายค่าเฉลี่ยของ Cross-validation สามารถเขียนได้ดังนี้:

$$\mathrm{CV}(k) = \frac{1}{k}\sum_{i=1}^{k} \mathrm{Score}_i$$

โดยที่:

- CV(k) คือค่า Cross-validation สำหรับค่า k ที่กำหนด (จำนวน fold)
- k คือจำนวน fold ที่ใช้ใน Cross-validation
- Score_i คือค่าประสิทธิภาพของแบบจำลองในรอบที่ i ของการทดสอบ (ในแต่ละ fold)

การทำ Cross-validation ช่วยให้เราได้ค่าประสิทธิภาพที่มีความน่าเชื่อถือมากขึ้นเมื่อมีการใช้ข้อมูลที่มีขนาดจำกัด และช่วยในการเลือกแบบจำลองที่มีประสิทธิภาพสูงสุดสำหรับงานที่กำลังดำเนินอยู่โดยประเมินได้ทั้งความแม่นยำและความเสถียรของแบบจำลองและช่วยป้องกัน overfitting ในการฝึกแบบจำลองด้วยด้วยการประเมินความสามารถในการทำนายในข้อมูลที่ไม่เคยเห็นมาก่อน.

Algorithm

Pseudo-code

```
# Pseudo-code for the k-Nearest Neighbors (k-NN) algorithm

# Input: training_data, test_instance, k

# Output: predicted_class

Set distances as an empty list

# Calculate distances between test_instance and each training_data point

For each training_point in training_data:

    dist = calculate_distance(training_point, test_instance)

    Add (training_point, dist) to distances

End For

Set flag to false

# Sort distances in ascending order

Sort distances by dist

# Select the top k points with the smallest distances

nearest_neighbors = distances[:k]
```

```
# Create a dictionary to store class votes

Set class_votes as an empty dictionary

# Count the votes for each class among the nearest_neighbors

For each neighbor in nearest_neighbors:

    class = neighbor.class  # Get the class of the neighbor

    If class is not in class_votes:

        Set class_votes[class] to 1

    Else:

        Increment class_votes[class] by 1

    End If

End For


# Find the class with the highest vote

Set predicted_class as the class with the highest count in class_votes

Return predicted_class
```

ผลการทดลอง

ผลการทดลองที่ใช้ dataset iris และใช้ 10% cross validation ในแต่ละค่า k ตั้งแต่ 1-19

```
k = 1, Average Accuracy: 0.9417
k = 2, Average Accuracy: 0.9417
k = 3, Average Accuracy: 0.9500
k = 4, Average Accuracy: 0.9250
k = 5, Average Accuracy: 0.9333
k = 6, Average Accuracy: 0.9417
k = 7, Average Accuracy: 0.9417
k = 8, Average Accuracy: 0.9417
k = 9, Average Accuracy: 0.9417
k = 10, Average Accuracy: 0.9417
k = 11, Average Accuracy: 0.9500
k = 12, Average Accuracy: 0.9417
k = 13, Average Accuracy: 0.9417
k = 14, Average Accuracy: 0.9333
k = 15, Average Accuracy: 0.9333
k = 16, Average Accuracy: 0.9333
k = 17, Average Accuracy: 0.9333
k = 18, Average Accuracy: 0.9167
k = 19, Average Accuracy: 0.9250
```

การวิเคราะห์ผลการทดลอง

การทดลองที่ใช้ dataset iris และ 10% cross-validation สำหรับค่า k ตั้งแต่ 1-19 ได้ผลลัพธ์ดังนี้:

k = 3, Average Accuracy: 0.9500: ค่า k เท่ากับ 3 มีความแม่นยำเฉลี่ยประมาณ 95.00% ซึ่งเริ่มแสดงให้เห็นว่าค่า k ที่ใหญ่ขึ้นอาจจะมีประสิทธิภาพดีกว่า.

k = 11, Average Accuracy: 0.9500: ค่า k เท่ากับ 11 มีความแม่นยำเฉลี่ยประมาณ 95.00% แสดงให้เห็นว่าค่า k ที่เพิ่มขึ้นมีประสิทธิภาพดีขึ้น.

k = 19, Average Accuracy: 0.9250: ค่า k เท่ากับ 19 มีความแม่นยำเฉลี่ยประมาณ 92.50% ลดลงเมื่อเทียบกับค่า k = 3 และ k = 11.

จากผลการทดลองนี้สามารถสรุปได้ว่าค่า k ที่ดีที่สุดสำหรับ dataset iris และการใช้ 10% cross-validation คือ k = 3 และ k = 11 โดยมีความแม่นยำเฉลี่ยสูงสุดที่ประมาณ 95.00%. การเลือกค่า k ที่เหมาะสมสำคัญในการปรับปรุงประสิทธิภาพของอัลกอริทึม k-NN.

ภาคผนวก

- Code

```
"""
File: kNN.py
Author: Chayadon Lappabud
Date: September 15, 2023
Description: Building a k-Nearest-Neighbors and testing on iris dataset.
Version: 1.0
"""
# Import libraries
import numpy as np
import random as rd # Only used for shuffling dataset

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

# Function calculate Distance
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2)**2))

# Function for sorting distances.
def argSort(distance:list)->list:
    sort_index = [i for i, x in sorted(enumerate(distance), key=lambda x: x[1])]
    return sort_index

# Function to find the most frequent element in a list.
def most_frequent(List):
    return max(set(List), key = List.count)

# k-Nearest Neighbors algorithm
def k_nearest_neighbors(X_train, y_train, x_query, k):
    distances = [euclidean_distance(x_query, x) for x in X_train]
    sorted_indices = argSort(distances)
    k_indices = sorted_indices[:k]
    k_nearest_labels = [y_train[i] for i in k_indices]
    most_common = most_frequent(k_nearest_labels)
    return most_common

# Cross Validation
def cross_validation(X, y, k_folds, k):
    fold_size = len(X) // k_folds
    accuracy_scores = []

    for i in range(k_folds):
        start, end = i * fold_size, (i + 1) * fold_size
        X_val_fold = X[start:end]
        y_val_fold = y[start:end]
```

```
        X_train_fold = np.concatenate((X[:start], X[end:]), axis=0)
        y_train_fold = np.concatenate((y[:start], y[end:]), axis=0)

        correct_predictions = 0
        for j in range(len(X_val_fold)):
            predicted_label = k_nearest_neighbors(X_train_fold, y_train_fold,
X_val_fold[j], k)
            if predicted_label == y_val_fold[j]:
                correct_predictions += 1

        accuracy = correct_predictions / len(X_val_fold)
        accuracy_scores.append(accuracy)

    avg_accuracy = np.mean(accuracy_scores)
    return avg_accuracy

# Split dataset into  training set and test set
def hold_out(train, test, num_train = 80):
    length = len(train)
    d = length*num_train // 100
    X_train = train[:d]
    X_test = test[:d]

    y_train = train[d:]
    y_test = test[d:]

    return X_train, X_test, y_train, y_test

# Shuffle the dataset
def fisher_yates_shuffle(arr):
    rd.shuffle(arr)

# Define a 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


#-----------------------------MAIN CODE-----------------------------------------------#
# Read the dataset from the specified file path
dataset =
read_data('F:/Advanced_Pattern_Recognition/Computer_assignment2.1/dataset/iris.pat')

# Set a seed for random shuffling
rd.seed(1337)

# Shuffle the dataset using Fisher-Yates shuffle
fisher_yates_shuffle(dataset)

# Define the number of folds for cross-validation
k_folds = 10

# Split the dataset into data and labels
X, y = train_test_split(dataset)
X = np.array(X)
y = np.array(y)
```

```
# Split the dataset into training and test sets
X_train, X_test, Y_train, Y_test = hold_out(X, y)

# Perform k-fold cross-validation for different values of k
for k in range(1, 20, 1):
    avg_accuracy = cross_validation(np.array(X_train), np.array(X_test), k_folds, k)
    print(f"k = {k}, Average Accuracy: {avg_accuracy:.4f}")
```

อ้างอิง

Data Innovation and Governance Institute. (2566). สรุปให้!! วิธีการคำนวนด้วย K-Nearest Neighbors. สืบค้นเมื่อ 9/17/2566, จาก https://digi.data.go.th/blog/what-is-k-nearest-neighbors/

Kasidis Satangmongkol. (2562). อธิบาย K-Fold Cross Validation พร้อมโค้ดตัวอย่างใน R. สืบค้นเมื่อ 9/17/2566, จาก https://datarockie.com/blog/k-fold-cross-validation/

Kasidis Satangmongkol. (2566). Confusion Matrix คืออะไร พร้อมวิธีคำนวณค่าสถิติง่ายๆ. สืบค้นเมื่อ 9/17/2566,จาก https://datarockie.com/blog/confusion-matrix-explained/#:~:text=Confusion%20matrix%20คือตาราง%20cross,cross%20กันตามชื่อเลย