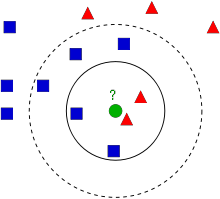
**Understanding k-Nearest Neighbour**

เป้าหมาย

ในบทนี้เราจะเข้าใจแนวคิดของ k-Nearest Neighbor (kNN) algorithm

ทฤษฎี

kNN เป็นหนึ่งในขั้นตอนวิธีการจำแนกประเภทที่ง่ายที่สุดสำหรับการเรียนรู้ที่ได้รับการดูแล ความคิดคือการค้นหาข้อมูลที่ใกล้เคียงที่สุดของข้อมูลทดสอบในพื้นที่ว่าง เราจะมองเข้าไปในภาพด้านล่าง



ในภาพมีสองครอบครัวสี่เหลี่ยมสีฟ้าและสีแดงรูปสามเหลี่ยม เราเรียกแต่ละครอบครัวเป็นชั้น บ้านของพวกเขาจะแสดงในแผนที่เมืองของพวกเขาซึ่งเราเรียกพื้นที่คุณลักษณะ (คุณสามารถพิจารณาพื้นที่คุณลักษณะเป็นพื้นที่ที่มีการฉายข้อมูลทั้งหมดตัวอย่างเช่นพิจารณาพื้นที่พิกัด 2D ข้อมูลแต่ละแห่งมีสองคุณสมบัติพิกัด x และ y คุณสามารถแสดงข้อมูลนี้ในพื้นที่พิกัด 2D ของคุณได้หรือไม่ตอนนี้ จินตนาการถ้ามีสามคุณสมบัติคุณต้องพื้นที่ 3D ตอนนี้พิจารณาคุณสมบัติ N ที่คุณต้องการพื้นที่ N - มิติใช่ไหมพื้นที่ N - นี้เป็นพื้นที่คุณลักษณะในภาพของเราคุณสามารถพิจารณาเป็นกรณี 2D กับ สองคุณสมบัติ)

ตอนนี้สมาชิกใหม่เข้ามาในเมืองและสร้างบ้านใหม่ซึ่งแสดงเป็นวงกลมสีเขียว เขาควรจะเพิ่มครอบครัวสีฟ้า / แดงเหล่านี้ เราเรียกว่ากระบวนการการจำแนกประเภท เราทำอะไร? เนื่องจากเรากำลังติดต่อกับ kNN ให้เราใช้อัลกอริธึมนี้

วิธีหนึ่งคือการตรวจสอบว่าใครเป็นเพื่อนบ้านที่ใกล้ที่สุดของเขา จากภาพที่เห็นได้ชัดก็คือครอบครัว Red Triangle ดังนั้นเขาจึงถูกเพิ่มลงในสามเหลี่ยมสีแดง วิธีนี้เรียกว่าเพียงเพื่อนบ้านที่ใกล้ที่สุดเนื่องจากการจำแนกประเภทขึ้นอยู่กับเพื่อนบ้านที่ใกล้ที่สุดเท่านั้น

แต่มีปัญหากับเรื่องนี้ สามเหลี่ยมสีแดงอาจอยู่ใกล้ที่สุด แต่ถ้ามีบลูสแควร์อยู่ใกล้เขามากสักนิด แล้วสี่เหลี่ยมสีฟ้ามีความแข็งแรงมากขึ้นในท้องถิ่นที่กว่าสามเหลี่ยมสีแดง ดังนั้นการตรวจสอบที่ใกล้ที่สุดไม่เพียงพอ แต่เราตรวจสอบkครอบครัวที่ใกล้ที่สุด แล้วใครก็ตามที่เป็นส่วนใหญ่ในพวกเขาคนที่แต่งตัวประหลาดใหม่เป็นของครอบครัวที่ ในภาพเราใช้เวลาk = 3ได้แก่ 3 ครอบครัวที่อยู่ใกล้ที่สุด เขามีสองแดงและหนึ่งสีฟ้า (มีสองบลูส์เท่ากัน แต่ตั้งแต่ k = 3 เราใช้เวลาเพียงหนึ่งของพวกเขา) ดังนั้นอีกครั้งเขาควรจะเพิ่มให้กับครอบครัวสีแดง แต่ถ้าเราใช้เวลาk = 7 ? แล้วเขาก็มีครอบครัวสีฟ้า 5 ครอบครัวและครอบครัวสีแดง 2 คน ยิ่งใหญ่ !! ตอนนี้เขาควรจะเพิ่มครอบครัว Blue ดังนั้นการเปลี่ยนแปลงทั้งหมดที่มีค่าของ k สิ่งที่ตลกมากคืออะไรถ้าk = 4? เขามีเพื่อนบ้าน 2 คนสีแดงและสีน้ำเงิน 2 คน มันเป็นเน็คไท !!! ดีกว่าให้ k เป็นเลขคี่ ดังนั้นวิธีนี้เรียกว่าk-Nearest Neighborเนื่องจากการจำแนกขึ้นอยู่กับ k เพื่อนบ้านที่อยู่ใกล้ที่สุด

อีกครั้งใน kNN เป็นเรื่องจริงที่เรากำลังพิจารณา k เพื่อนบ้าน แต่เราให้ความสำคัญเท่าเทียมกันกับทุกคนใช่มั้ย? มันยุติธรรม? ตัวอย่างเช่นใช้กรณีของk = 4 เราบอกว่ามันเป็นเน็คไท แต่ครอบครัวสีแดง 2 ครอบครัวใกล้ชิดกับเขามากขึ้นกว่าครอบครัวสีน้ำเงินอีก 2 ครอบครัว ดังนั้นเขามีสิทธิ์มากขึ้นที่จะถูกเพิ่มลงในสีแดง ดังนั้นวิธีการที่เราทางคณิตศาสตร์อธิบายว่า? เราให้น้ำหนักกับแต่ละครอบครัวขึ้นอยู่กับระยะทางของพวกเขาไปยังผู้มาใหม่ สำหรับผู้ที่อยู่ใกล้เขาจะมีน้ำหนักที่สูงขึ้นขณะที่อยู่ห่างไกลจะได้รับน้ำหนักที่ต่ำกว่า จากนั้นเราจะเพิ่มน้ำหนักรวมของแต่ละครอบครัวแยกกัน ใครก็ตามที่มีน้ำหนักรวมสูงที่สุดผู้ที่มาใหม่จะไปที่ครอบครัวนั้น นี่เรียกว่าkNN ที่แก้ไขแล้ว

ดังนั้นสิ่งที่สำคัญบางอย่างที่คุณเห็นที่นี่?

คุณจำเป็นต้องมีข้อมูลเกี่ยวกับบ้านทั้งหมดในเมืองใช่ไหม? เพราะเราต้องตรวจสอบระยะห่างจากผู้มาใหม่มายังบ้านที่มีอยู่ทั้งหมดเพื่อหาเพื่อนบ้านที่ใกล้ที่สุด หากมีบ้านและครอบครัวมากมายจะใช้เวลามากของหน่วยความจำและมีเวลามากขึ้นสำหรับการคำนวณด้วย

เกือบจะเป็นศูนย์เวลาสำหรับการฝึกอบรมหรือการเตรียมการใด ๆ

ตอนนี้ขอดูใน OpenCV

kNN ใน OpenCV

เราจะทำตัวอย่างง่ายๆที่นี่กับสองครอบครัว (ชั้นเรียน) เช่นเดียวกับข้างต้น จากนั้นในบทถัดไปเราจะทำตัวอย่างที่ดียิ่งขึ้น

ดังนั้นที่นี่เราจะให้ชื่อว่า Red family เป็นClass-0 (แสดงเป็น 0) และ Blue เป็นClass-1 (แสดงด้วย 1) เราสร้าง 25 ครอบครัวหรือ 25 ข้อมูลการฝึกอบรมและตั้งชื่อว่า Class-0 หรือ Class-1 เราทำสิ่งเหล่านี้ด้วยความช่วยเหลือของเครื่องกำเนิดตัวเลขสุ่มใน Numpy

จากนั้นเราวางแผนด้วยความช่วยเหลือของ Matplotlib ครอบครัวสีแดงแสดงเป็นรูปสามเหลี่ยมสีแดงและครอบครัวสีน้ำเงินจะแสดงเป็นสี่เหลี่ยมสีฟ้า

**import** **cv2**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

*# Feature set containing (x,y) values of 25 known/training data*

trainData = np.random.randint(0,100,(25,2)).astype(np.float32)

*# Labels each one either Red or Blue with numbers 0 and 1*

responses = np.random.randint(0,2,(25,1)).astype(np.float32)

*# Take Red families and plot them*

red = trainData[responses.ravel()==0]

plt.scatter(red[:,0],red[:,1],80,'r','^')

*# Take Blue families and plot them*

blue = trainData[responses.ravel()==1]

plt.scatter(blue[:,0],blue[:,1],80,'b','s')

plt.show()

คุณจะได้รับสิ่งที่คล้ายกับภาพแรกของเรา เนื่องจากคุณใช้เครื่องคำนวณหมายเลขแบบสุ่มคุณจะได้รับข้อมูลที่แตกต่างกันทุกครั้งที่คุณเรียกใช้โค้ด

ถัดไปเริ่มต้นอัลกอริทึม kNN และส่งข้อมูลtrainDataและการตอบสนองไปยังการฝึกอบรม kNN (สร้างโครงสร้างการค้นหา)

จากนั้นเราจะนำคนที่มาใหม่คนหนึ่งมาจัดกลุ่มให้ครอบครัวหนึ่งคนด้วยความช่วยเหลือของ kNN ใน OpenCV ก่อนที่จะไปที่ kNN เราจำเป็นต้องทราบข้อมูลบางอย่างในข้อมูลทดสอบของเรา (ข้อมูลของผู้เดินทางใหม่) จำนวน \;  ของ \;  หมายเลข testdata \ times \;  ของ \;  คุณสมบัติข้อมูลของเราควรจะเป็นอาร์เรย์ลอยจุดที่มีขนาด จากนั้นเราจะหาเพื่อนบ้านที่ใกล้ที่สุดของผู้มาใหม่ เราสามารถระบุจำนวนเพื่อนบ้านที่เราต้องการได้ จะส่งกลับ:

ป้ายกำกับให้กับผู้มาใหม่ขึ้นอยู่กับทฤษฎี kNN ที่เราได้เห็นมาก่อนหน้านี้ ถ้าคุณต้องการอัลกอริทึมเพื่อนบ้านที่ใกล้ที่สุดเพียงแค่ระบุk = 1โดย k คือจำนวนเพื่อนบ้าน

ป้ายชื่อ k-Neighbours เพื่อนบ้าน

ระยะห่างที่สอดคล้องกันจากผู้มาใหม่กับเพื่อนบ้านที่ใกล้ที่สุด

ลองดูวิธีการทำงาน ผู้มาใหม่มีเครื่องหมายสีเขียว

newcomer = np.random.randint(0,100,(1,2)).astype(np.float32)

plt.scatter(newcomer[:,0],newcomer[:,1],80,'g','o')

knn = cv2.KNearest()

knn.train(trainData,responses)

ret, results, neighbours ,dist = knn.find\_nearest(newcomer, 3)

**print** "result: ", results,"**\n**"

**print** "neighbours: ", neighbours,"**\n**"

**print** "distance: ", dist

plt.show()

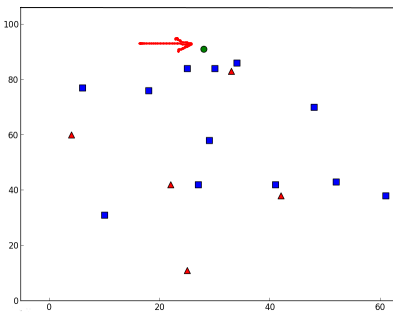
ผมได้ผลดังนี้

result: [[ 1.]]

neighbours: [[ 1. 1. 1.]]

distance: [[ 53. 58. 61.]]

บอกว่าเพื่อนใหม่ของเรามีเพื่อนบ้าน 3 คนจากครอบครัว Blue ดังนั้นเขาจึงมีชื่อว่า Blue family เป็นที่ชัดเจนจากพล็อตด้านล่าง:



หากคุณมีข้อมูลจำนวนมากคุณสามารถส่งผ่านข้อมูลนี้เป็นอาร์เรย์ ผลลัพธ์ที่ได้จะได้รับเป็นอาร์เรย์

*# 10 new comers*

newcomers = np.random.randint(0,100,(10,2)).astype(np.float32)

ret, results,neighbours,dist = knn.find\_nearest(newcomer, 3)

*# The results also will contain 10 labels.*