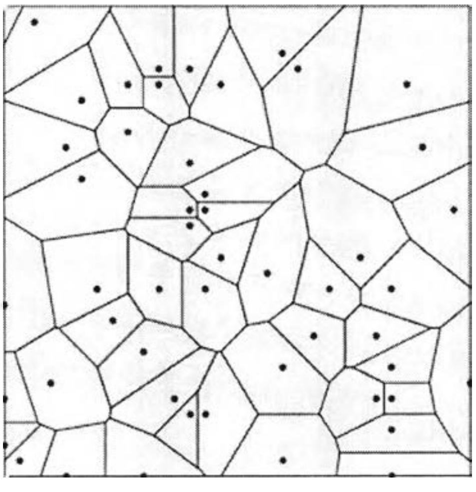
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 แผนภาพโวโรนอย

แผนภาพโวโรนอย (Voronoi diagram) [1,2,3,4,5] สามารถอธิบายในทางคณิตศาสตร์ได้ว่าเป็นการแบ่งส่วนอาณาบริเวณที่มีให้เป็นรูปทรงหลายเหลี่ยม (polygon) หลายๆ อัน โดยในรูปทรงหลายเหลี่ยมแต่ละอันจะมีจุดกำเนิด (generating point) อยู่ 1 จุด ซึ่งเป็นจุดที่ทำให้เกิดรูปทรงหลายเหลี่ยมอันนั้นขึ้นมา โดยมีเงื่อนไขที่ว่าจุดใดๆ ก็ตามที่อยู่ภายในรูปทรงหลายเหลี่ยมแต่ละอัน จะต้องอยู่ใกล้กับจุดกำเนิดที่อยู่ภายในรูปทรงหลายเหลี่ยมอันนั้นมากกว่าจุดกำเนิดจุดอื่น ซึ่งสามารถอธิบายให้เข้าใจได้ง่ายๆ ว่า แผนภาพโวโรนอยเป็นแผนภาพที่แสดงการแบ่งปริภูมิเป็นส่วนๆ ตามจุดกลุ่มหนึ่งที่เป็นข้อมูลเบื้องต้น โดยในการแบ่งนั้นมีข้อกำหนดอยู่ว่าจุดที่อยู่ในเซลล์หรือช่องใดๆ ของแผนภาพโวโรนอยจะต้องมีระยะทางใกล้กับจุดซึ่งเป็นข้อมูลเบื้องต้นในเซลล์นั้นมากกว่าจุดเบื้องต้นจุดอื่น นั่นก็คือเป็นเหมือนกับแผนภาพที่แสดงผลของเนียเรสท์เนเบอร์ในกรณีที่ k เป็น 1 กล่าวคือเป็นการแบ่งบริเวณโดยอาศัยข้อมูลที่มีความใกล้เคียงที่สุดเพียงตัวเดียว โดยเซลล์นั้นก็เป็นเหมือนกับการบอกว่าผลของการจำแนกประเภทจะถูกนำมาจากข้อมูลเบื้องต้นตัวใด ซึ่งขนาดของเซลล์นั้นก็เป็นเหมือนกับการบอกว่าข้อมูลเบื้องต้นที่อยู่ในบริเวณนั้นมีความหนาแน่นเพียงใด



รูปที่ 1 ตัวอย่างแผนภาพโวโรนอยใน 2 มิติ จุดที่ปรากฏเป็นเหมือนกับตัวแทนของข้อมูลตัวอย่างในอัลกอริทึมเนียเรสท์เนเบอร์ พื้นที่ของเซลล์แสดงถึงบริเวณที่ข้อมูลตัวอย่างที่อยู่ภายในนั้นถูกใช้เป็นผลของการจำแนกประเภทสำหรับอัลกอริทึมเนียเรสท์เนเบอร์เมื่อ k=1

มีการคิดค้นอัลกอริทึมสำหรับใช้ในการคำนวณขนาดของเซลล์ของแผนภาพโวโรนอยจากข้อมูลเบื้องต้น [6] และมีโปรแกรมสำเร็จรูปอยู่หลายชิ้นที่สามารถทำการคำนวณหาขนาดของเซลล์ของแผนภาพโวโรนอยได้โดยหนึ่งในโปรแกรมที่รู้จักกันดีคือ Qhull [7] ซึ่งได้ชื่อมาจากอัลกอริทึมควิกฮัล (quickhull) [8] ที่โปรแกรมนี้ใช้สำหรับหาคอนเวกซ์ฮัล (convex hull) ซึ่งเป็นรูปทรงหลายเหลี่ยมที่ไม่มีการเว้า (convex polygon) ที่เล็กที่สุดที่ครอบคลุมจุดพิกัดที่เป็นข้อมูลทั้งหมด โดยโปรแกรม Qhull นี้เป็นโปรแกรมที่โปรแกรม Matlab จะทำการเรียกใช้อีกทีสำหรับงานจำพวกการหาคอนเวกซ์ฮัลและการสร้างแผนภาพโวโรนอย

จากการที่มีอัลกอริทึมต่างๆ และมีโปรแกรมสำหรับใช้คำนวณย่อมแสดงว่าเรามีวิธีการที่เป็นแบบแผนที่จะใช้ในการคำนวณหาขนาดของเซลล์ของแผนภาพโวโรนอยแต่ละเซลล์ได้ อย่างไรก็ตาม ปัญหาสำคัญของอัลกอริทึมเหล่านี้ก็คือเวลาที่ใช้จะเพิ่มขึ้นตามจำนวนมิติของปัญหา ซึ่งแม้แต่โปรแกรม Qhull ที่ได้ยกตัวอย่างไปในข้างต้น ซึ่งจัดได้ว่ามีประสิทธิภาพในการทำงานดีแล้วแต่ในทางปฏิบัติก็ยังสามารถทำงานได้อย่างมีประสิทธิภาพเฉพาะเมื่อมีจำนวนมิติน้อย ๆ เท่านั้น นั่นก็คือสำหรับข้อมูลที่มีจำนวนมิติมากๆ นั้นถึงแม้ว่าเราจะมีวิธีการที่ใช้คำนวณได้แต่เนื่องจากเวลาที่ใช้นั้นมากเกินไปจึงไม่เหมาะสมที่จะนำมาใช้ในทางปฏิบัติ

2.1.2 การตรวจสอบไขว้

การตรวจสอบไขว้ k พับ (k-fold cross validation) เป็นวิธีการหนึ่งที่ใช้สำหรับเพิ่มความแม่นยำในการวัดค่าความถูกต้องของวิธีการจำแนกประเภทหนึ่งๆ บนข้อมูลที่ทดสอบ ในการทดสอบประสิทธิภาพของการจำแนกประเภทนั้น โดยปกติแล้วเราจะมีข้อมูลอยู่จำนวนหนึ่งซึ่งเราทราบประเภทอยู่แล้ว ข้อมูลเหล่านี้จะถูกแบ่งออกเป็น 2 ส่วน โดยส่วนหนึ่งจะถูกนำไปเป็นข้อมูลสอน (training data) เพื่อใช้ในการสร้างตัวจำแนกประเภท ขณะที่ส่วนที่เหลือจะถูกนำมาใช้เป็นข้อมูลทดสอบ (test data) เพื่อวัดความถูกต้อง (accuracy) ของตัวจำแนกประเภทที่ได้ จะเห็นได้ว่าหากทำเช่นนี้แล้วอาจจะเกิดความคลาดเคลื่อนของค่าความถูกต้องวัดได้จากการแบ่งข้อมูลตัวอย่างสำหรับฝึกสอนและการทดสอบเพราะในการแบ่งกลุ่มนั้นข้อมูลที่ถูกเลือกไปอยู่ในกลุ่มหนึ่งอาจจะมีลักษณะเฉพาะตัวบางอย่างซึ่งเมื่อนำไปเปรียบเทียบกับข้อมูลที่เหลือแล้วจะทำให้ผลของการวัดค่าความถูกต้องคลาดเคลื่อนไปได้มาก และนอกจากนั้นแล้วก็จะมีปัญหาเกิดขึ้นได้จากการแบ่งข้อมูลไปยังประเภทใดประเภทหนึ่งมากเกินไป เพราะถ้าหากเป็นเช่นนั้นแล้วก็จะทำให้ข้อมูลที่อยู่ในอีกประเภทหนึ่งมีจำนวนลดลง โดยหากลองพิจารณาดูจะได้ว่า ถ้าหากข้อมูลที่ใช้ฝึกสอนมีน้อยจะทำให้ตัวจำแนกประเภทที่ได้มีความแม่นยำต่ำ นั่นก็คืออัลกอริทึมที่ใช้อาจจะยังทำงานได้ไม่เต็มประสิทธิภาพก่อนที่จะถูกนำไปทดสอบ และในทางตรงกันข้าม ถ้าข้อมูลทดสอบมีน้อยเกินไปก็จะทำให้ค่าความถูกต้องวัดได้มีความแปรปรวนสูงเนื่องมากจากการที่ข้อมูลที่ใช้วัดผล

นั้นมีจำนวนน้อยจะทำให้ค่าความถูกต้องที่วัดได้จากการเฉลี่ยมีความน่าจะเป็นที่จะคลาดเคลื่อนค่อนข้างสูง นั่นก็คือผลที่ได้จากการทดลองอาจจะแตกต่างจากความเป็นจริงไปมาก

เราสามารถที่จะแก้ปัญหาเหล่านี้ได้หลายวิธีโดยวิธีการหนึ่งซึ่งเป็นที่นิยมใช้สำหรับลดผลกระทบจากปัญหาเหล่านี้ก็คือการตรวจสอบไขว้ k พับ โดยในการทำงานของวิธีการนี้นั้น เมื่อเริ่มต้น ข้อมูลตัวอย่างจะถูกแบ่งออกเป็น k พับหรือ k ส่วนซึ่งมีจำนวนข้อมูลในแต่ละส่วนเท่าๆ กัน หลังจากนั้นในการทำงานแต่ละรอบก็จะทำการเลือกข้อมูลออกมา 1 พับเพื่อนำมาใช้ในการทดสอบ แล้วนำข้อมูลในพับที่เหลือทั้งหมดมารวมกันเพื่อที่จะใช้ในเป็นข้อมูลสอน และเมื่อทำเช่นนี้กับทุกพับเสร็จ แล้วท้ายที่สุดก็จะนำค่าความถูกต้องที่ได้ทั้งหมดมาเฉลี่ยกัน จะเห็นได้ว่าจำนวนข้อมูลที่ใช้ในการทดสอบจะมีเท่ากับจำนวนข้อมูลทั้งหมด และจำนวนข้อมูลที่ใช้ในการฝึกสอนก็สามารถปรับได้ตามค่าของ k ขณะเดียวกันผลจากการเลือกแบ่งข้อมูลในพับต่างๆ จะลดลงเนื่องจากจำนวนข้อมูลสอนที่มากขึ้นจะทำให้ข้อมูลสอนสะท้อนลักษณะของข้อมูลโดยรวมได้ดียิ่งขึ้น ซึ่งสาเหตุเหล่านี้เองทำให้ผลการทดลองที่ได้มีความถูกต้องค่อนข้างสูง อย่างไรก็ตามวิธีการทดสอบแบบไขว้นี้จะส่งผลให้ต้องทำการฝึกสอนและทดสอบเป็นจำนวนมากขึ้น นั่นก็คือนี่เป็นวิธีการที่จะใช้เวลามาชดเชยกับความแม่นยำของผลการทดลองแต่ก็จะลดปัญหาเรื่องการตัดสินใจในการแบ่งกลุ่มข้อมูลลงไปได้ เพราะฉะนั้นจึงเป็นวิธีการที่เหมาะที่จะนำไปใช้เมื่อมีจำนวนข้อมูลที่จะใช้ในการทดสอบค่อนข้างน้อย

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 อัลกอริทึมเคเนียเรสท์เนเบอร์ [1,2]

ก่อนอื่นเราจะอธิบายการทำงานของอัลกอริทึมเนียเรสท์เนเบอร์ดั้งเดิมหรือที่มักจะเรียกกันว่าเคเนียเรสท์เนเบอร์อีกครั้ง โดยเราสามารถที่จะอธิบายอัลกอริทึมเป็นรหัสเทียมแบบง่ายๆ ได้ดังต่อไปนี้

D = a set of training data

p = the data to be classified

k = the number of used nearest neighbors

$n(X)$ = the number of members of a set X

```
K-Nearest Neighbor(D,p,k)
    H ← {}
    For each (d ∈ D)
        If (n(H) < k)
            H ← H ∪ {d}
        Else
            m ← arg max_{x∈H} distance (p, x)
            If (distance (p, d) < distance (p, m))
                H ← (H − {m}) ∪ {d}
            End If
        End If
    End For
    For ( i = 1  to the number of possible classes)
        class_i ← 0
    End For
    For each (d ∈ H)
        r ← class(d)
        class_r ← class_r + 1
    End For
    Return  arg max_i (class_i)
```

รหัสเทียมดังกล่าวจะแสดงขั้นตอนการทำงานของอัลกอริทึมเคเนียเรสท์เนเบอร์ในการจำแนกประเภทข้อมูลเพียงครั้งเดียว โดยจากในรหัสเทียมจะเห็นได้ว่าอัลกอริทึมนี้จะมีอินพุทอยู่ทั้งหมด 3 ตัว อันได้แก่ D เป็นเซตของข้อมูลตัวอย่างทั้งหมด p คือข้อมูลที่ต้องการจำแนกประเภท และ k เป็นจำนวนของข้อมูลตัวอย่างที่คล้ายคลึงที่ต้องการใช้สำหรับจำแนกประเภทกับข้อมูลที่พิจารณา ส่วนรายละเอียดของอัลกอริทึมนั้นก็เป็นไปตังที่ได้กล่าวไปแล้วในข้างต้นโดยจะเห็นได้จากในรหัสเทียมที่แสดงอยู่

2.2.2 **อัลกอริทึมเนียเรสท์เนเบอร์แบบถ่วงน้ำหนัก** [1,2]

กรณีที่มีการใช้ฟังก์ชันถ่วงน้ำหนักก็จะทำการเปลี่ยนแปลงอัลกอริทึมโดยการเพิ่มส่วนที่คำนวณค่าของฟังก์ชันลงไปและเปลี่ยนผลของการเพิ่มค่าที่ใช้สำหรับพิจารณาผลลัพธ์จากที่เพิ่มด้วย 1 สำหรับข้อมูลทุกตัวเป็นการเพิ่มด้วยค่าที่ได้จากฟังก์ชันถ่วงน้ำหนักแทน โดยเราสามารถแสดงอัลกอริทึมเนียเรสท์เนเบอร์แบบถ่วงน้ำหนักเป็นรหัสเทียมได้ดังต่อไปนี้

D = a set of training data

p = the data to be classified

f(p,d,D) = weight function

```
Weighted-Nearest Neighbor(D,p,f)
    For ( i = 1 to the number of possible classes)
        class_i ← 0
    End For
    For each (d ∈ D)
        r ← class(d)
        class_r ← class_r + f(p,d,D)
    End For
    Return arg max_i (class_i)
```

อัลกอริทึมนี้จะมีอินพุทอยู่สองตัวเช่นเดียวกับอัลกอริทึมเคเนียเรสท์เนเบอร์ได้แก่ D ซึ่งเป็นเซตของข้อมูลตัวอย่างทั้งหมดและ p คือข้อมูลที่ต้องการจำแนกประเภท แต่ว่าแทนที่จะใช้ค่า k ซึ่งเป็นจำนวนข้อมูลที่มีลักษณะใกล้เคียงกันซึ่งถูกใช้ในการจำแนกประเภทของเคเนียเรสท์เนเบอร์ ก็จะใช้ฟังก์ชัน f ซึ่งเป็นฟังก์ชันถ่วงน้ำหนักสำหรับการให้คะแนนแทน

สำหรับอัลกอริทึมเนียเรสท์เนเบอร์แบบถ่วงน้ำหนักนั้น ฟังก์ชันถ่วงน้ำหนักที่เป็นพื้นฐานที่สุดซึ่งเป็นที่นิยมใช้ก็คือฟังก์ชัน $f(x,y)=\dfrac{1}{d(x,y)^2}$ โดยฟังก์ชันนี้จะรับอินพุท 2 ค่าคือ x ซึ่งเป็นข้อมูลสอนและ y ซึ่งเป็นจุดที่ต้องการจำแนกประเภท โดย $d(x,y)$ เป็นฟังก์ชันระยะทางยูคลิดซึ่งจะให้ผลลัพธ์เป็นระยะทางยูคลิดระหว่างจุด x และ y

ฟังก์ชันนี้มีลักษณะที่น่าจะใช้เป็นฟังก์ชันถ่วงน้ำหนักได้จากการที่เมื่อระยะทางระหว่างจุดทั้งสองนั้นน้อยมากๆ เราจะได้ว่าค่าของฟังก์ชันนี้จะเข้าใกล้ ∞ นั่นหมายความถ้าข้อมูลนั้นมีระยะทางระหว่างกันเป็น 0 หรือเป็นข้อมูลเดียวกันก็น่าจะใช้ค่าจากข้อมูลนั้นเป็นคำตอบได้เลยหากไม่มีความผิดพลาดของข้อมูล และการที่ค่าของฟังก์ชันเข้าใกล้ 0 เมื่อระยะทางระหว่างกันมีค่ามากๆ นั่นคือข้อมูลสอนตัวนั้นจะไม่มีผลต่อการจำแนกประเภท และเมื่อระยะทางระหว่างกันมีค่ามากขึ้นก็จะได้ว่าค่าของฟังก์ชันจะลดลงนั่นคือจะมีผลต่อการจำแนกประเภทลดลงไปตามความแตกต่างซึ่งวัดได้โดยระยะทาง

สำหรับการใช้ค่ายกกำลังสองนั้น เนื่องจากการคำนวณระยะทางยูคลิดจะอาศัยการยกกำลังสองของค่าในแต่ละมิติมาบวกรวมกันแล้วจึงนำค่าที่ได้มาถอดรากที่สองเป็นคำตอบ ด้วยเหตุนี้การใช้ระยะทางยูคลิดยกกำลังสองจึงเป็นการยกเลิกการหาค่ารากที่สองในตอนท้ายซึ่งจะทำให้การหาค่าของระยะทางยูคลิดยกกำลังสองนั้นใช้เวลาน้อยกว่าการหาค่าระยะทางยูคลิด

นอกจากนี้จะเห็นได้ว่าฟังก์ชัน $f(x,y)=\frac{1}{d(x,y)^2}$ จะทำการหารเพียงครั้งเดียวหลังจากที่หาค่าของระยะทางยูคลิดยกกำลังสองเสร็จแล้ว ซึ่งถือว่าเป็นฟังก์ชันที่ง่ายที่สุดหลังจากหาค่าของระยะทางยูคลิดยกกำลังสองที่จะมีค่าเข้าใกล้ 0 เมื่อ $d(x,y)=\infty$ เนื่องจากการปฏิบัติการทางคณิตศาสตร์ที่ง่ายกว่าหรือเท่ากับการหารของคอมพิวเตอร์นั้นมีเพียงการบวก การลบ และ การคูณเท่านั้น โดยจะเห็นได้ว่าการบวกกับลบค่า ∞ ด้วยค่าคงที่ใดๆ จะให้ผลลัพธ์เป็น ∞ หรือ $-\infty$ เท่านั้น และค่าเดียวที่คูณเข้าไปกับค่าที่เข้าใกล้ ∞ แล้วได้ 0 ก็มีแต่ 0 ซึ่งการคูณเข้าไปจะทำให้ผลลัพธ์จากฟังก์ชันนั้นมีค่าเป็น 0 สำหรับอินพุทใดๆก็ตาม นั่นก็คือไม่มีฟังก์ชันถ่วงน้ำหนักที่อาศัยระยะทางยูคลิดที่ใช้การคำนวณน้อยไปกว่านี้อีกแล้ว

ฟังก์ชันถ่วงน้ำหนักอื่นที่ใช้กันและมีความยุ่งยากกว่าฟังก์ชันที่กล่าวมาเล็กน้อยก็คือฟังก์ชันในรูปแบบ $f(x,y)=\frac{1}{d(x,y)^t};t>0$ ซึ่งก็คือการใช้ฟังก์ชันถ่วงน้ำหนักที่ใช้ผลจากการหาร 1 ด้วยระยะทางยูคลิดยกกำลัง $t/2$ ซึ่งโดยปกติแล้วจะใช้ค่าระยะทางยูคลิดยกกำลังด้วยค่าที่มากกว่า 2 ขึ้นไป

อย่างไรก็ตามเนื่องจากฟังก์ชันนี้จะมีค่าเป็น ∞ เมื่อจุดที่ทำการพิจารณาอยู่เป็นจุดเดียวกัน นั่นก็คือถ้ามีข้อมูลที่มีค่าของคุณสมบัติตรงกันหมดก็จะใช้ค่าจากข้อมูลนั้นเป็นคำตอบได้เลยโดยสมมุติฐานที่ข้อมูลไม่มีความคลาดเคลื่อน แต่ถ้าหากสมมุติว่าในข้อมูลที่ได้รับมามีข้อมูลสอนอยู่หลายตัวที่มีค่าของคุณสมบัติทุกประการตรงกับข้อมูลที่ต้องการจำแนกประเภท เพื่อให้ผลที่เป็นคำตอบของการจำแนกประเภทมีความถูกต้องสูงสุดเราจึงควรจะใช้คำตอบเป็นเสียงส่วนใหญ่ของข้อมูลสอนเหล่านั้น ซึ่งเมื่อทำการแก้ไขตรงจุดนี้เข้าไปก็จะได้อัลกอริทึมดังต่อไปนี้

```
D = a set of training data
p = the data to be classified
f(p,d,D) = weight function
Exact Data Handled Weighted-Nearest Neighbor(D,p,f)
    For (i = 1 to the number of possible classes)
        class_i ← 0
        exactclass_i ← 0
    End For
    For each (d ∈ D)
        r ← class(d)
        If (f(p,d,D)= ∞ )
            exactclass_r ← exactclass_r +1
        Else
            class_r ← class_r + f(p,d,D)
        End If
    End For
    s ← arg max_i (exactclass_i)
```

If ($exactclass_s \neq 0$)
return s
End If
Return $\arg\max_i (class_i)$

เมื่อเราพิจารณาอัลกอริทึมเคเนียเรสท์เนเบอร์เปรียบเทียบกับการใช้อัลกอริทึมเนียเรสท์เนเบอร์แบบถ่วงน้ำหนัก เราจะสังเกตได้ว่าการเลือกข้อมูลที่มีความใกล้เคียงกับข้อมูลที่ต้องการจำแนกประเภทที่สุดจำนวนหนึ่งก็ถือได้ว่าเป็นฟังก์ชันการให้คะแนนหรือถ่วงน้ำหนักของข้อมูลรูปแบบหนึ่งเช่นกัน โดยเป็นการให้น้ำหนักข้อมูลที่ถูกเลือกด้วยจำนวนจริงบวกค่าหนึ่งขณะที่ให้คะแนนของข้อมูลที่เหลือทั้งหมดเท่ากับศูนย์ นั่นก็คืออัลกอริทึมเนียเรสท์เนเบอร์แบบถ่วงน้ำหนักนี้เป็นกรณีทั่วไปเคเนียเรสท์เนเบอร์

2.2.3 ฟังก์ชันระยะทาง

โดยปกตินั้นเวลาเราพูดถึงคำว่าระยะทางระหว่างจุดเราจะหมายถึงระยะห่างระหว่างพิกัดในปริภูมิยูคลิด ซึ่งระยะทางยูคลิดนี้สามารถคำนวณได้โดยกฎคณิตศาสตร์พื้นฐานของปีทากอรัส อย่างไรก็ตามก็มีผู้คนที่ได้ให้การบัญญัติระยะทางชนิดอื่นๆขึ้นมามากมาย โดยถึงแม้จะไม่ใช่ระยะทางที่คำนวณในโลกแห่งความเป็นจริงอย่างระยะทางยูคลิดซึ่งสมมุติให้โลกเป็นไปตามกลศาสตร์ของนิวตัน แต่โดยหลักแล้วการตั้งระยะทางแต่ละชนิดนั้นจะขึ้นอยู่กับความต้องการที่จะนำไปใช้ โดยการบัญญัติขึ้นมานั้นมักจะขึ้นกับความต้องการที่จะนำไปใช้สอยในงานแต่ละประเภท เราสามารถสรุปประเภทของระยะทางต่างๆ ที่รู้จักกันทั่วไปและแสดงฟังก์ชันระยะทางซึ่งเป็นฟังก์ชันที่รับอินพุท 2 ค่าคือจุดพิกัดทั้งสองแล้วให้คำตอบออกมาเป็นระยะทางระหว่างพิกัดตามชนิดของระยะทางนั้นได้ดังต่อไปนี้

1. **ระยะทางยูคลิด** (Euclidean distance) [5,9,10,11,12] **เป็นระยะทางที่**กล่าวถึงเมื่อใช้คำว่าระยะทางในความหมายทั่วไป ซึ่งสามารถคำนวณได้โดยกฎของปีทากอรัส สามารถเขียนเป็นสมการได้ว่า $c^2 = a^2 + b^2$ เมื่อ c เป็นความยาวของด้านตรงข้ามมุมฉาก และ a กับ b เป็นด้านประกอบมุมฉากทั้งสอง ซึ่งถ้าให้ a และ b เป็นมิติสองมิติ ก็จะได้ว่า c เป็นระยะทางยูคลิดใน 2 มิติ และเมื่อเราทำการขยายผลไปโดยการเพิ่มจำนวนมิติที่สูงขึ้นเป็น 3 มิติ โดยให้ g เป็นมิตินั้นและ h เป็นระยะทางยูคลิดในปริภูมิยูคลิดที่มี 3 มิติ โดยอาศัยกฎของปีทากอรัสเราจะได้สมการว่า $h^2 = g^2 + c^2 = g^2 + a^2 + b^2$ ซึ่งสามารถอธิบายเป็นคำพูดได้ว่าระยะทางยูคลิดระหว่างจุดพิกัดใดๆ ยกกำลังสอง จะมีค่าเท่ากับผลรวมของระยะทางยูคลิดในแต่ละมิติยกกำลังสอง

ด้วยวิธีการเดียวกันนี้เอง เราสามารถขยายวิธีการคำนวณระยะทางยูคลิดไปยังปริภูมิยูคลิดที่มีจำนวนมิติสูงขึ้นได้ และเมื่อทำการจัดรูปสมการเราจะได้ฟังก์ชันระยะทาง

ยูคลิดเป็น $d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}$ เมื่อกำหนดให้ $d(x,y)$ เป็นฟังก์ชันระยะทางซึ่งรับอินพุทสองค่าคือ x และ y ที่เป็นจุดพิกัดทั้งสอง โดยมี m เป็นจำนวนมิติ และให้ x_i และ y_i เป็นค่าพิกัดของ x และ y ในมิติที่ i

2. ระยะทางแมนฮัตตัน (Manhattan distance) [5,9,10,11,12,13] หรือที่เรียกอีกอย่างหนึ่งว่าระยะทางแบบบล็อกของเมือง (city-block distance) หรือ หน่วยวัดระยะของรถแท็กซี่ (taxicab metric) ระยะทางประเภทนี้จะใช้การคำนวณแบบความห่างกันของสิ่งปลูกสร้างว่าห่างกันกี่บล็อกในเมืองอย่างแมนฮัตตันซึ่งเป็นเขตพื้นที่ในเมืองที่พื้นที่ถูกแบ่งออกเป็นบล็อกสี่เหลี่ยมผืนผ้าขนาดเท่าๆกัน เรียงต่อกันไปเรื่อยๆทั่วทั้งเมือง เช่น ถ้าสถานที่แห่งหนึ่งอยู่ห่างจากที่ที่เราสนใจไปทางตะวันออก 3 บล็อกและต้องขึ้นไปทางเหนืออีก 4 บล็อก ก็จะได้ว่าสถานที่แห่งนั้นอยู่ออกจากที่แห่งนี้ออกไปเป็นระยะทาง=3+4=7 บล็อก

จะเห็นได้ว่าการคำนวณระยะแบบนี้ย่อมไม่ถูกต้องสำหรับปริภูมิยูคลิด แต่ว่าการคำนวณระยะทางแบบแมนฮัตตันนั้นใช้ประโยชน์ได้ โดยจะสังเกตได้ว่าการที่จะเดินทางในเมืองประเภทนี้จะโดยอาศัยยานพาหนะทางบกหรือการเดินเท้า จะต้องไปตามถนนซึ่งแบ่งเมืองออกเป็นบล็อกต่างๆ ดังนั้นการรู้ว่าสถานที่ที่เป็นเป้าหมายนั้นอยู่ห่างจากที่ปัจจุบันนี้ไป 5 ช่วงถนน หรือ 5 บล็อกซึ่งแปรผันตรงตามระยะทางที่ต้องใช้ในจริงในการสัญจรไปตามเส้นทางบนพื้นถนน จึงจะเป็นประโยชน์ต่อการเดินทางมากกว่าการรู้ระยะทางยูคลิดว่าจุดที่สนใจนั้นอยู่ห่างออกไปอีก 300 เมตร

เราสามารถจะเขียนฟังก์ชันระยะทางแมนฮัตตันออกมาเป็นสมการได้ว่า $d(x,y)=\sum_{i=1}^{m}|x_i-y_i|$ ซึ่งจะเห็นได้ว่าข้อดีข้อหนึ่งของฟังก์ชันนี้เมื่อเปรียบเทียบกับฟังก์ชันระยะทางยูคลิดก็คือการคำนวณจะทำได้ง่ายกว่าเนื่องจากในการคำนวณจะใช้คำสั่งของเครื่องในการบวกลบเท่านั้น ขณะที่การคำนวณระยะทางยูคลิดจะต้องใช้คำสั่งสำหรับคูณเพื่อทำการยกกำลังสองแล้วยังต้องใช้การหารากที่สองอีกด้วย

3. ระยะทางเชบีเชฟ (Chebyshev distance) [10,11] หรือระยะทางแบบตารางหมากรุก (chessboard distance) เราสามารถเขียนฟังก์ชันระยะทางสำหรับระยะทางประเภทนี้ได้ว่า $d(x,y)=\max_{i\in[1,m]}(|x_i-y_i|)$ ซึ่งอธิบายได้ว่าเป็นการใช้ค่าสัมบูรณ์ของความแตกต่างระหว่างค่าในแนวพิกัดในมิติที่มีความแตกต่างกันมากที่สุดมาเป็นคำตอบ วิธีการหนึ่งที่นิยมเอามาใช้แสดงถึงระยะทางเชบีเชฟก็คือการเดินของตัวคิงบนหมากรุกฝรั่ง โดยในการเกมหมากรุกฝรั่งซึ่งเป็นเกมตารางขนาด 8 คูณ 8 ที่ใช้การผลัดกันเดินตัวหมากฝ่ายละตัวในแต่ละรอบนั้นจะมีหมากตัวหนึ่งที่เรียกว่าคิงซึ่งแปลว่าพระราชา หมากตัวนี้มีลักษณะการเดินแบบเดียวกับตัวหมากที่เรียกว่าขุนใน

หมากรุกไทย โดยสามารถจะเดินไปทุกช่องที่ติดกันได้รวมทั้งในแนวทแยง นั่นก็คือกรณีที่อยู่บริเวณที่โล่งๆ บนกระดาน หมากตัวนี้จะสามารถเลือกเดินไปได้ยังช่องใดก็ได้ช่องหนึ่งในทั้ง 8 ช่องที่ติดกัน ซึ่งได้แก่ ช่องหน้า หลัง ซ้าย ขวา หน้าซ้าย หน้าขวา หลังซ้าย และหลังขวาตามลำดับ คำถามก็คือถ้าต้องการให้คิงตัวนี้ซึ่งอยู่บนกระดานโล่งๆ เดินไปยังช่องว่างช่องหนึ่งบนกระดาน คิงตัวนี้จะต้องเดินทั้งหมดกี่ครั้ง

เนื่องจากกระดานนี้มี 2 มิติ ตามแนวตั้ง (หน้าหลัง) และแนวขวาง (ซ้ายขวา) และการเดินในแต่ละตาของคิงจะสามารถทำให้ความแตกต่างระหว่างตำแหน่งปัจจุบันกับตำแหน่งที่เป็นเป้าหมายลดลงไปได้ 1 จากการที่เมื่อสิ้นสุดการเดินความแตกต่างระหว่างตำแหน่งปัจจุบันกับช่องที่เป็นเป้าหมายจะต้องเป็น 0 นั่นก็คือ หากกำหนดให้ $s(a,b)$ เป็นฟังก์ชันจำนวนครั้งที่คิงตัวนี้จะต้องเดินที่รับอินพุทเป็นค่าสัมบูรณ์ที่แสดงความแตกต่างกันของช่องในแนวตั้งและแนวขวาง เราจะได้ว่า

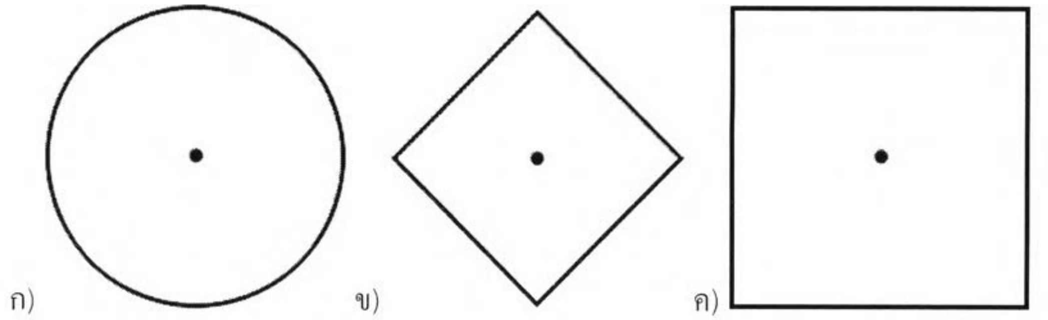
$$s(a,b) = s(a-1,b-1); a > 0; b > 0$$

$$s(a,0) = s(a-1,0); a > 0$$

$$s(0,b) = s(0,b-1); b > 0$$

และ $s(0,0) = 0$

เราสามารถทำการพิสูจน์ด้วยวิธีการอุปนัย (induction) โดยให้ 3 สมการแรกเป็นขั้นตอนอุปนัย (inductive step) และให้สมการสุดท้ายเป็นขั้นตอนฐานหลัก (basis step) เป็นการยืนยันได้ว่า $s(a,b) = \max(a,b); a \geq 0, b \geq 0$ นั่นก็คือคิงตัวนี้จะต้องใช้จำนวนครั้งในการเดินรวมทั้งสิ้นเท่ากับความห่างกันของช่องในมิติที่ห่างกันมากกว่า นั่นเอง และเมื่อทำการแทนค่าของ a และ b ด้วย $|x_1 - y_1|$ และ $|x_2 - y_2|$ เราจะได้ว่า $s(a,b) = \max_{i \in [1,2]}(|x_i - y_i|)$ ซึ่งเป็นตรงกับฟังก์ชันระยะทางแบบเชบีเชฟใน 2 มิติ หรือก็คือการจำนวนครั้งที่คิงจะต้องขยับจะมีค่าเท่ากับระยะทางแบบเชบีเชฟระหว่างช่องที่เป็นจุดเริ่มต้นกับจุดหมาย



รูปที่ 2 ผลของฟังก์ชันระยะทางแบบต่างๆ บนปริภูมิ 2 มิติ โดยจุดตรงกลางแทนจุดที่สนใจ ส่วนเส้นรอบนอกแสดงบริเวณที่มีระยะห่างจากจุดศูนย์กลางเท่ากันเป็นระยะ 1 หน่วย
ก) ระยะทางยูคลิด ข) ระยะทางแมนฮัตตัน ค) ระยะทางเชบีเชฟ

4. ระยะทางมินโคสกี (Minkowski distance) [11] ระยะทางประเภทนี้สามารถเขียนอธิบายออกมาเป็นรูปสมการได้ว่า $d(x,y)=\left(\sum_{i=1}^{m}|x_i-y_i|^r\right)^{1/r}$ นั่นก็คือการใช้ค่าสัมบูรณ์ของความแตกต่างกันของพิกัดในแต่ละมิติมายกกำลังด้วยค่าคงที่บวก r ค่าหนึ่งก่อนที่จะนำมาบวกเข้าด้วยกันแล้วหารากที่ r ของผลรวมมาใช้เป็นคำตอบ จะเห็นได้ว่าเมื่อให้ค่าปรับแต่ง $r=2$ ฟังก์ชันนี้จะกลายฟังก์ชันระยะทางของระยะทางยูคลิดและหากว่าเรากำหนดให้ $r=1$ ฟังก์ชันนี้จะกลายฟังก์ชันระยะทางของระยะทางแมนฮัตตัน และนอกจากนี้หาก $r=\lim_{x\to\infty} x$ ก็จะได้ว่า $\left(\frac{|x_{i-}y_i|}{|x_{j-}y_j|}\right)^r=0$ เมื่อ $j=\arg\max_{i\in[1,n]}(|x_i-y_i|)$ หรือ j เป็นคุณสมบัติที่ทำให้ $|x_j-y_j|$ มีค่ามากที่สุด และ $i\neq j$ หรืออีกนัยหนึ่งก็คือ ผลลัพธ์จากคุณสมบัติอื่นๆ ที่ไม่ใช่ j จะไม่มีผลต่อผลลัพธ์ และหากมีคุณสมบัติที่ให้ค่า $|x_i-y_i|$ มากที่สุดมากกว่า 1 ตัว โดยให้มีจำนวนเท่ากับ k เราก็จะได้ว่า $d(x,y)=(k|x_j-y_j|^r)^{1/r}=(k^{1/r})(|x_j-y_j|)=|x_j-y_j|$ ซึ่งเมื่อแทนค่าของ j ลงไปสมการนี้ก็จะเปลี่ยนไปเป็น $d(x,y)=\max_{i\in[1,m]}(|x_i-y_i|)$ ซึ่งเป็นสมการของระยะทางเชบีเชฟ นั่นก็คือเมื่อ $r=\lim_{x\to\infty} x$ ฟังก์ชันนี้จะกลายฟังก์ชันระยะทางของระยะทางเชบีเชฟ

ระยะทางมินโคสกีนี้จะแตกต่างจากระยะทางยูคลิด ระยะทางแมนฮัตตัน และระยะทางเชบีเชฟตรงที่เป็นการยากกว่าที่จะอธิบายถึงประโยชน์ของมันในโลกของความเป็นจริงแต่ว่าระยะทางแบบนี้จะเป็นกรณีทั่วไปที่จำเพาะที่สุดของระยะทางทั้งสามแบบที่ได้กล่าวถึงมาแล้ว

5. ระยะทางควอดราติก (quadratic distance) [11] ฟังก์ชันระยะทางประเภทนี้สามารถแสดงออกมาเป็นสมการได้ว่า $d(x,y)=(x-y)^T A(x-y)$ เมื่อ A เป็นเมตริกซ์ขนาด $m\times m$ ซึ่งเป็นค่าปรับแต่งที่มักจะเป็นลักษณะของปัญหาที่ต้องแก้และ x และ y เป็นเวกเตอร์

ขนาด m ของจุดพิกัดทั้งสองซึ่งก็คือเมตริกซ์ขนาด $m \times 1$ ซึ่งหากทำการแตกค่าจากสมการนั้นออกมาจะได้ว่า $d(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij}(x_i - y_i)(x_j - y_j)$ โดยปกติแล้ว ฟังก์ชันประเภทนี้มักจะปรากฏเป็นคะแนนของปัญหาประเภทที่กำหนดเมตริกซ์ A มาให้แล้วต้องการให้หาคู่ของ x และ y ซึ่งทำให้ระยะทางนี้มีค่ามากสุดหรือน้อยสุด หรืออาจกำหนดตัวใดตัวหนึ่งมาด้วยแล้วต้องการให้หาค่าของอีกตัว เช่น กำหนด A และ x แล้วต้องการให้หาค่าของ y

จะเห็นว่าฟังก์ชันประเภทนี้จะต่างกับประเภทที่ได้กล่าวถึงมาแล้วโดยระยะทางประเภทนี้จะมีการคูณค่ากันระหว่างค่าที่ได้มาจากคนละมิติ นั่นก็คือความสัมพันธ์กันของค่าในมิติที่ต่างกันจะส่งผลต่อการการคำนวณด้วย และสังเกตด้วยว่าจากการที่ไม่ใช้ค่าสัมบูรณ์และไม่ได้กำหนดเครื่องหมายของ a_{ij} ดังนั้นผลที่ได้ในแต่ละก้อนที่จะเอามาบวกกันจึงอาจส่งผลต่อผลลัพธ์ได้ทั้งในทางบวกและทางลบ และผลลัพธ์ที่ได้ก็สามารถที่จะติดลบได้ด้วย

ข้อสังเกตอีกประการหนึ่งก็คือ หากทำการกำหนดค่าของ A โดยกำหนดให้

$$a_{ij} = 1; i = j$$

$$a_{ij} = 0; i \neq j$$

จะเห็นได้ว่าสมการนี้จะเปลี่ยนรูปให้ง่ายขึ้นได้โดยการตัดนิพจน์ที่เป็นผลคูณกันของค่าคนละมิติออกทำให้ได้ว่า ที่เป็นได้เป็น $d(x, y) = \sum_{i}^{m} (x_i - y_i)^2$ ซึ่งก็คือค่าของระยะทางยูคลิดยกกำลังสอง

นอกจากรูปแบบต่างๆที่ได้กล่าวถึงไปแล้วยังมีรูปแบบของฟังก์ชันระยะทางในรูปแบบอื่นๆอีกหลายชนิด ยกตัวอย่างเช่น ระยะทางแคนเบอร์รา (Canberra distance) ซึ่งเราสามารถเขียนเป็นสมการได้ว่า $d(x, y) = \sum_{i}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$ โดยจะเป็นการหาความแตกต่างของค่าในแต่ละมิติเมื่อทำการคิดเทียบความแตกต่างนั้นเป็นสัดส่วนของค่าตัวของตัวมันเอง ซึ่งแม้จะมีการคิดค้นฟังก์ชันหรือวิธีคำนวณระยะทางขึ้นมาหลายรูปแบบแต่ฟังก์ชันนำไปใช้กันมากที่สุดก็ยังเป็นฟังก์ชันของระยะทางยูคลิด โดยฟังก์ชันระยะทางที่สำคัญๆ ที่เหลือนั้นก็มีดังที่ได้แนะนำไปแล้ว