โปรแกรมรู้จำเสียง

Voice Recognition

นาย อรรณพ แสงเสนาะ

นาย อิทธินันท์ คงเกียรติก้อง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

โปรแกรมรู้จำเสียง

Voice Recognition

โดย

นาย อรรณพ แสงเสนาะ

นาย อิทธินันท์ คงเกียรติก้อง

อาจารย์ที่ปรึกษา

ผศ.เกียรติกูล เจียรนัยธนะกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมรู้จำเสียง

Voice Recognition

ผู้จัดทำ

1. นาย อรรณพ แสงเสนาะ เลขประจำตัว 45010938

2. นาย อิทธินันท์ คงเกียรติก้อง เลขประจำตัว 45010975

อาจารย์ที่ปรึกษา

(ผศ.เกียรติกูล เจียรนัยธนะกิจ)

โปรแกรมรู้จำเสียง

นาย อรรณพ แสงเสนาะ 45010938
นาย อิทธินันท์ คงเกียรติก้อง 45010975
ผศ. เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2548

บทคัดย่อ

ในปัจจุบันนี้ ได้มีอุปกรณ์ที่ใช้ในการรับอินพุทมากมาย เช่น คีย์บอร์ด จอยสติ๊ก จอแบบทัชสกรีน แต่ในกรณีที่ต้องการเพิ่มความสะดวกสบายให้ผู้ใช้งาน หรือ เพื่อรองรับผู้ใช้งานที่มีความพิการทางร่างกาย จึงได้มีการนำเอาเสียงมนุษย์มาใช้ในการติดต่อกับคอมพิวเตอร์

ปริญญานิพนธ์ฉบับนี้จะเป็นการศึกษาเกี่ยวกับการนำเสียงของมนุษย์มาใช้ติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ผ่านทางไมโครโฟนแทนการใช้คีย์บอร์ด แล้วนำสัญญาณเสียงที่ได้เข้าสู่การ์ดเสียงเพื่อนำมาประมวลผลในโปรแกรม จากนั้นจึงใช้โครงข่ายประสาทเทียมแบบย้อนกลับ เป็นระบบที่ใช้ในการเรียนรู้คำสั่งต่างๆ 10 คำสั่ง และเมื่อผู้ใช้พูดคำสั่งที่ได้เรียนรู้ไปแล้ว โปรแกรมก็จะให้ผลลัพท์ว่าเสียงที่พูดนั้นคือเสียงอะไร แล้วนำผลลัพท์นั้นไปประยุกต์ใช้ในขั้นต่อไป

Voice Recognition

Unnop Sangsanor 45010001

Ittinun Kongkiatkong 45010001

Kietikul Jearanaitanakij Advisor

Academic Year 2005

ABSTRACT

At the present time, There are various types of input device, such as keyboards, joysticks, touch-screen monitors that provide users in many ways of using computers. But in case of more convenience or disability users. Many researchers consider about applying human voice for connecting with computers.

The purpose of this thesis is to introduce human voice for controlling computers via a microphone instead of a keyboard device and bring voice signal through sound card in order to execute in the program. Back-propagation Neural network is used to learn any commands about ten commands in number. When users speak those commands which are learned. The program will return the result what command users say and execute that command.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ คำปรึกษาและคอยดูแลจากหลาย ๆ ฝ่ายด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาที่คอยให้ความเอาใจใส่ แนะนำและช่วยเหลือเสมอมา ซึ่งก็คือ อาจารย์เกียรติกูล เจียรนัยธนะกิจ ขอขอบพระคุณเป็นอย่างสูง

นอกเหนือจากนี้ต้องขอขอบพระคุณอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเป็นอย่างยิ่งที่ได้ช่วยประสิทธิ์ประสาทวิชาความรู้ให้แก่คณะผู้จัดทำ

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เช่น อินเทอร์เน็ตความเร็วสูง ซึ่งช่วยให้การวิจัย การค้นคว้าหาความรู้ต่าง ๆ และพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวกและรวดเร็ว

สุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดาและบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ ความรักเสมอมา ข้าพเจ้าขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

อรรณพ แสงเสนาะ

อิทธินันท์ คงเกียรติก้อง

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 บทนำ

โดยปกติแล้วการติดต่อสื่อสารระหว่างมนุษย์กับเครื่องคอมพิวเตอร์ทำได้โดยการป้อนคำสั่งผ่านทางคีย์บอร์ดและเมาส์ ในขณะที่ต้องการหาวิธีการอื่นที่เป็นไปอย่างสะดวกและเป็นธรรมชาติมากกว่า จึงมีการพัฒนาให้คอมพิวเตอร์สามารถรับรู้คำสั่งจากเสียงพูดของมนุษย์ได้ ซึ่งจะทำให้การติดต่อสื่อสารระหว่างมนุษย์และคอมพิวเตอร์เป็นไปอย่างสะดวกและง่ายขึ้น

สมัยก่อนการใช้คอมพิวเตอร์เพื่อรับรู้เสียงพูดของมนุษย์และวิเคราะห์เสียงพูดนั้นทำได้ยากและใช้เวลานาน เนื่องจากการพูดของมนุษย์มีความซับซ้อนและมีความแตกต่างในแต่ละบุคคลทำให้การพัฒนาเป็นไปอย่างล่าช้า แต่ในปัจจุบันได้มีการพัฒนาวิธีการต่างๆ ให้มีความสามารถในการที่จะหาตัวแทนของเสียงในรูปแบบต่าง ๆ รวมทั้งมีวิธีการเพิ่มมากขึ้น และมีความแม่นยำในการรู้จำเสียงที่ป้อนให้แก่คอมพิวเตอร์ ด้วยพื้นฐานความรู้ทางด้านคณิตศาสตร์และพื้นฐานความรู้ทางด้านสถิติจึงทำให้เราลดความยุ่งยากลงได้มากจนการสั่งงานคอมพิวเตอร์ด้วยเสียงแทนการใช้คีย์บอร์ดและเมาส์เริ่มเข้ามามีบทบาทในการสื่อสารระหว่างมนุษย์กับคอมพิวเตอร์มากขึ้น ซึ่งเพิ่มความสะดวกสบายให้แก่ผู้ใช้ และสามารถนำมาช่วยคนพิการ อีกทั้งยังใช้ในงานรักษาความปลอดภัยโดยการระบุผู้พูด (Speaker Recognition) ได้อีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาการทำงานของระบบรู้จำเสียง

1.2.2 เพื่อศึกษาการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการ Microsoft Windows

1.2.3 เขียนโปรแกรมประยุกต์เพื่อพัฒนาระบบรู้จำเสียงมาใช้งานจริง

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ ความเข้าใจในหลักของการรู้จำเสียงของมนุษย์ และคอมพิวเตอร์

1.3.2 ได้รับความรู้ในการวิเคราะห์รูปแบบเสียง

1.3.3 ได้รับความรู้ในการนำทฤษฎีนิวรอลเน็ตเวิร์คมาประยุกต์ใช้งาน

1.3.4 ได้รับความรู้ ความสามารถในการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการ Microsoft Windows

1.4 ขอบเขตของโครงงาน

1.4.1 พัฒนาโปรแกรมการเรียนรู้และทำงานระบบด้วยโปรแกรม Microsoft Visual C++ .NET พัฒนาด้วยรูปแบบ C#

1.4.2 พัฒนาระบบรู้จำเสียงที่ไม่จำกัดลักษณะคำสั่งของเจ้าของเครื่อง ในที่นี้เราทำการทดลองเพียง 10 คำสั่ง

1.5 เนื้อหาของรายงาน

รายงานนี้ประกอบด้วยเนื้อหา 5 บท โดย บทที่ 1 จะกล่าวถึงบทนำ วัตถุประสงค์ และขอบเขตของโครงงาน บทที่ 2 จะกล่าวถึงทฤษฎีทั้งหมดที่ใช้ในการทำโครงงาน ส่วนในบทที่ 3 จะกล่าวถึงการออกแบบและพัฒนาโปรแกรม บทที่4 แสดงผลการทดลองของระบบรู้จำเสียงโดยมีการวิเคราะห์หาค่าจุดตัดเสียงที่เหมาะสม จำนวนรูปแบบอ้างอิงและสภาวะแวดล้อมในการรู้จำเสียงที่ทำให้ระบบมีประสิทธิภาพในการรู้จำสูงที่สุด และบทที่ 5 จะเป็นการสรุปเนื้อหารายงานทั้งหมดและผลการดำเนินงาน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 บทนำ

ในปัจจุบันความก้าวหน้าทางด้านการประมวลผลสัญญาณต่าง ๆ ได้รุดหน้าไปอย่างรวดเร็ว ซึ่งในบางโอกาสสัญญาณอาจถูกประมวลผลในระบบอนาลอก (Analog) ในบางโอกาสก็ถูกประมวลผลในระบบดิจิตอลด้วยเครื่องคอมพิวเตอร์ ซึ่งในระบบของการประมวลผลด้วยเครื่องคอมพิวเตอร์ สัญญาณที่ถูกประมวลผลจะถูกแปลงจากสัญญาณอนาลอกที่ต่อเนื่องไปเป็นสัญญาณดิสครีท (Discrete) มีการสุ่มเอาตัวอย่างสัญญาณไปประมวลผล ซึ่งอยู่ในส่วนการทำงานของการ์ดเสียง จากนั้นจะเป็นขั้นตอนการประมวลผลซึ่งมีมากมายหลายวิธี ซึ่งล้วนแต่มีวัตถุประสงค์ในการประมวลผลสัญญาณเพื่อแยกแยกสัญญาณต่าง ๆ ไปทำประโยชน์ตามต้องการ

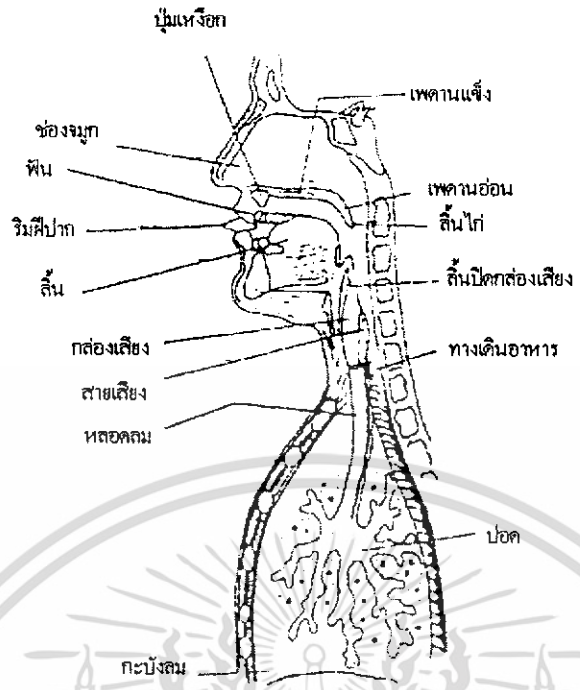
2.2 ศาสตร์ด้านการรู้จำ

ศาสตร์ด้านการรู้จำ หมายถึง การให้คอมพิวเตอร์รู้จักรูปแบบของสิ่งใดสิ่งหนึ่ง โดยเก็บบันทึกคุณลักษณะบางอย่างเพื่อให้สามารถบอกได้ว่าสิ่งนั้นๆ คืออะไร ซึ่งที่จริงแล้วมนุษย์เองไม่ได้ทราบถึงคุณลักษณะพิเศษเหล่านั้น ในการแยกแยะตามธรรมชาติ ศาสตร์ด้านการรู้จำในช่วงแรกๆ นั้นจะนำหลักสถิติมาใช้ แต่พบ มีขีดจำกัดค่อยข้างมากในการเลียนแบบการรู้จำของมนุษย์ เพราะสมอง ปัญญา และความฉลาดของมนุษย์ ไม่ใช่เรื่องของสถิติเพียงอย่างเดียว

2.3 การเปล่งเสียงของมนุษย์

จากการศึกษาด้านกายวิภาคศาสตร์ของมนุษย์ (human anatomy) วิชาที่ว่าด้วยเสียงของภาษา (phonetics) และศาสตร์ทางด้านเสียง (acoustics) ช่วยให้เข้าใจขั้นตอนการทำงานร่วมกันของอวัยวะต่างๆ
ในการเปล่งเสียงพูด ตลอดจนลักษณะทางกายภาพของเสียงพูดเพื่อนำมาวิเคราะห์ และสร้างแบบจำลองเลียนแบบเสียงพูดของมนุษย์

การทำให้เกิดเสียงเป็นหน้าที่หนึ่งของระบบหายใจ การออกเสียงหรือการพูดของมนุษย์แต่ละครั้ง จะต้องมีการทำงานร่วมกันของอวัยวะต่าง ๆ ของร่างกาย ดังรูปที่ 2.1 อันประกอบด้วย

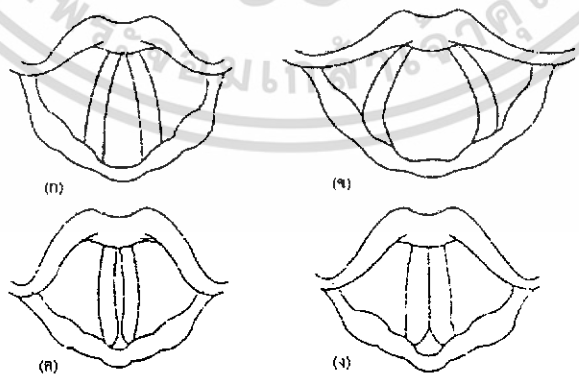


รูปที่ 2.1 ภาพตัดขวางแสดงอวัยวะในระบบการพูดของมนุษย์

1. ปอดและกระบังลม ทำหน้าที่สำคัญในการหายใจ และเป็นต้นกำเนิดการไหลของอากาศในกระบวนการผลิตเสียง

2. หลอดลม ทำหน้าที่นำอากาศจากปอดผ่านกล่องเสียง และเป็นอวัยวะที่อยู่ด้านหน้าของหลอดอาหาร

3. กล่องเสียง เป็นอวัยวะพิเศษที่ทำหน้าที่เป็นทางเดินอากาศเวลาหายใจ และเป็นตัวผลิตพัลส์ (pulse) ของอากาศขณะเปล่งเสียง ซึ่งประกอบด้วยเส้นเสียง (vocal cords) และช่องสายเสียง (glottis) รูปร่างของกล่องเสียง และเส้นเสียงในลักษณะต่าง ๆ แสดงในรูปที่ 2.2



รูปที่ 2.2 รูปกล่องเสียงขณะ

(ก) หายใจปกติ (ข) หายใจเข้าลึก ๆ

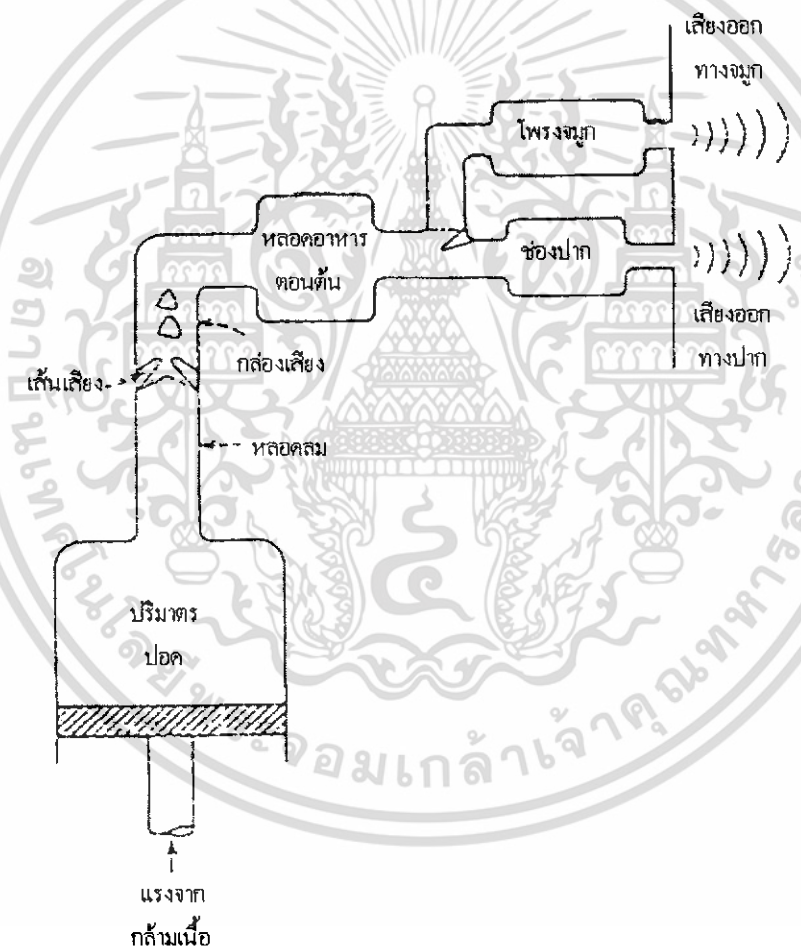
(ค) กำลังส่งเสียง (ง) ส่งเสียงกระซิบหรือเสียงแผ่ว

4. ช่องปากและส่วนของหลอดอาหารตอนต้น อวัยวะกลุ่มนี้อยู่ต่อจากกล่องเสียง อาจเรียกว่าอวัยวะกำทอนเสียง (vocal tract) ทำหน้าที่กำทอนเสียง โดยให้กำทอนทั้งเสียงที่เกิดจากกล่องเสียง และเสียงที่เกิดภายในช่องปาก ขนาดของอวัยวะกำทอนเสียงขึ้นอยู่กับตำแหน่งของลิ้น ริมฝีปาก ขากรรไกร และเพดานอ่อน และเปลี่ยนแปลงไปตามการออกเสียง

5. โพรงจมูก เริ่มจากเพดานอ่อนจนถึงรูจมูกทั้งสอง ทำหน้าที่กำทอนเสียงร่วมกับช่องปาก เมื่อมีการเปล่งเสียงที่ออกจากจมูก (nasal sounds) เช่นเสียง /ม/ , /น/ และ /ง/ เป็นต้น

2.4 กระบวนการผลิตเสียงพูด

จากระบบเสียงพูด สามารถแสดงเป็นแผนภาพของระบบกำเนิดเสียง ดังรูปที่ 2.3



รูปที่ 2.3 แผนภาพระบบเสียงพูดของมนุษย์

ซึ่งสามารถจำแนกกลไกการสร้างเสียงพูดของมนุษย์ได้ 3 แบบ ดังนี้

1. อากาศที่ไหลจากปอดจะถูกมอดูเลท (modulate) โดยการสั่งของเส้นเสียงทำให้เกิดคลื่นเสียงลักษณะคล้ายพัลส์ที่มีคาบเวลาแบบควอไซ (quasi-periodic pulse-like excitation)

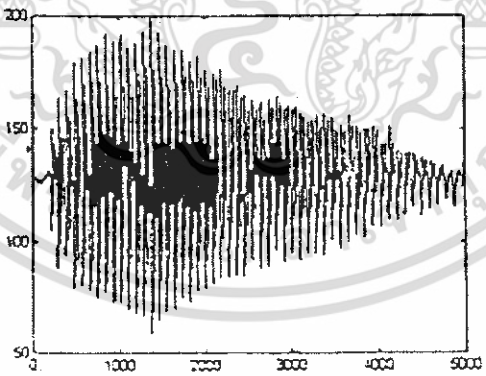
2. อากาศที่ไหลจากปอดถูกทำให้ปั่นป่วนด้วยการบังคับให้ไหลผ่านช่องแคบอันเกิดจากการบีบตัวของอวัยวะในช่องปากทำให้เกิดเสียงลักษณะคล้ายเสียงรบกวน (noise-like excitation)

3. อากาศที่ไหลถูกกัก และเกิดแรงดันอยู่ภายในส่วนของช่องปากที่ปิด จากนั้นจึงปล่อยให้อากาศที่มีแรงดันพุ่งออกไปอย่างเร็ว ทำให้เกิดการกระตุ้นเป็นเสียงในช่วงเริ่มต้น (transient excitation)

2.5 เสียงพูดของมนุษย์

เสียงพูดเป็นคลื่นตามยาว (longitudinal wave) เกิดจากการสั่นของอนุภาคตัวกลางนั่นคืออากาศ และทิศทางการสั่นของอนุภาค จะอยู่ในทิศเดียวกันกับทิศทางของการเคลื่อนที่ คลื่นเสียงเป็นคลื่นที่เปลี่ยนแปลงไปตามเวลา เสียงพูดแบ่งออกได้เป็น 2 ชนิด ตามการกำเนิดเสียง หรือโหมด (mode) การกระตุ้น คือ

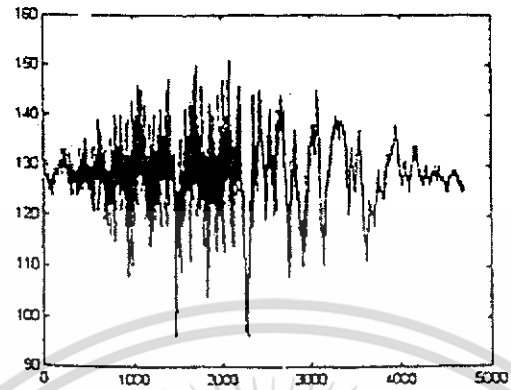
1. เสียงก้องหรือเสียงโฆษะ (voiced) เกิดจากการบังคับอากาศให้ผ่านช่องสายเสียงทำให้มีการเปลี่ยนแปลงความตึงหย่อนของเส้นเสียง โดยเส้นเสียงจะสั่นและเกิดเป็นพัลส์ (pulse) ของอากาศไปกระตุ้นอวัยวะกำทอนเกิดเป็นเสียงก้อง ตัวอย่างเสียงก้องได้แก่ เสียงสระ เสียงพยัญชนะ ที่ต้องออกเสียงจากลำคอ (voiced consonants) เช่น เสียง /a/ ดังรูป 2.4 (ก)



รูปที่ 2.4 (ก) ตัวอย่างรูปคลื่นของสัญญาณเสียงก้อง /a/

2. เสียงไม่ก้องหรืออโฆษะ (unvoiced หรือ voiceless) เป็นเสียงที่เกิดในช่องปาก หรือโพรงจมูก โดยอวัยวะภายในช่องปาก ริมฝีปาก ขวางการไหลของอากาศให้ผ่านได้เป็นช่องเล็ก ๆ อากาศจึงไหลผ่านอย่างรวดเร็ว และปั่นป่วนจนกระทั่งสร้างเป็นเสียงรบกวนช่องความถี่กว้าง (broad-

spectrum noise) ตัวอย่างเสียงไม่ก้องได้แก่ เสียงพยัญชนะที่ไม่ได้เกิดจากลำคอ (voiceless consonants) เช่น เสียง /sh/. ดังรูปที่ 2.4 (ข)



รูปที่ 2.4 (ข) ตัวอย่างรูปคลื่นของสัญญาณเสียงไม่ก้อง /sh/

2.6 แบบจำลองระบบกำเนิดเสียงพูด

จากภาพรูปที่ 2.5 สามารถแสดงภาพกรอบจำลองระบบกำเนิดเสียงเบื้องต้น โดยมีการแยกภาคแหล่งกำเนิดสัญญาณกระตุ้น ออกจากส่วนกำทอนเสียง ซึ่งแทนด้วยระบบเชิงเส้นแปรผันตามเวลา

รูปที่ 2.5 แผนภาพกรอบจำลองระบบกำเนิดเสียงเริ่มต้น

จากรูปที่ 2.5 แหล่งกำเนิดสัญญาณกระตุ้นทำหน้าที่แทนการทำงานของปอด และกล่องเสียงส่วนนี้จะผลิตขบวนพัลส์ที่มีคาบเวลาพิตช์ขณะเปล่งเสียง และให้กำเนิดเสียงซึ่งคล้ายเสียงรบกวนขณะเปล่งเสียงอโฆษะ (unvoiced) ส่วนที่สองเป็นท่อกำทอนเสียง จะแทนการทำงานของช่องปาก และโพรงจมูก ทำหน้าที่เสมือนตัวกรองสัญญาณ (filter) ที่ยอมให้ความถี่ฟอร์แมนท์ผ่านได้ ซึ่งสามารถแทนด้วยระบบเชิงเส้นแปรผันตามเวลา (time-varying linear system)

2.7 รูปแบบของไฟล์เสียง

ในการบันทึกเสียงพูดเป็นข้อมูลดิจิตอลโดยใช้การ์ดเสียง จะทำให้ได้เวฟไฟล์ที่มีรูปแบบของไฟล์ RIFF (Resource International File Format) ซึ่งเป็นรูปแบบมาตรฐานที่ใช้กันอย่างกว้างขวาง ไฟล์ RIFF นั้นมีโครงสร้างเชิงซ้อนคือ มีลักษณะเป็นกลุ่ม (Chunk) ซึ่งภายในกลุ่มนี้ยังประกอบด้วยกลุ่มย่อยซึ่งมีโครงสร้างคล้ายกัน ดังแสดงในรูปที่ 2.6

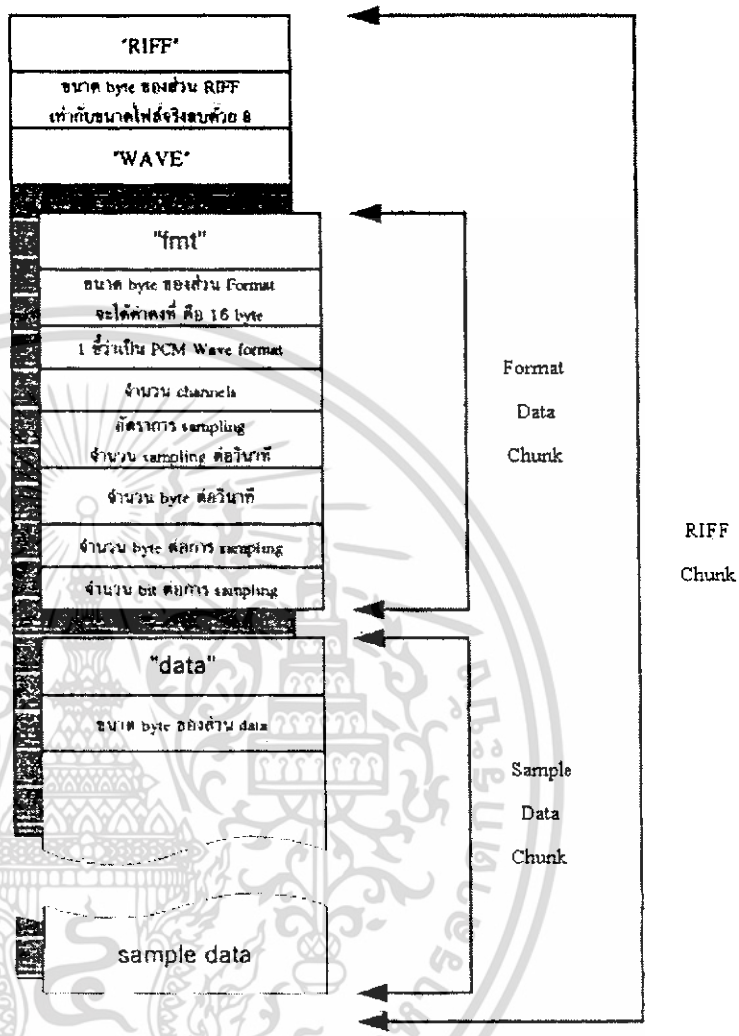
จากรูปที่ 2.6 จะเห็นว่า กลุ่ม RIFF (RIFF Chunk) จะเป็นกลุ่มใหญ่ที่สุด ซึ่งประกอบด้วยกลุ่มรูปแบบของข้อมูล (Format Data Chunk) และกลุ่มข้อมูลจากการแซมปลิ้ง (Sample Data Chunk) ซึ่งในแต่ละกลุ่มจะมีโครงสร้างที่ประกอบด้วย

1. ชื่อกลุ่ม (CKID : Chunk Identification) เช่น RIFF , fmt_ , data ซึ่งมีกฎกำหนดว่าต้องมี 4 ตัวอักษร (FCC : Four Character Code) ถ้ามี 3 ตัวอักษรก็จะต้องเพิ่มช่องว่าง (Blank) เข้าไปอีก 1 ตัวอักษรให้รวมเป็น 4 ตัวอักษรในกรณี fmt_

2. ตัวเลขบอกขนาดข้อมูลของกลุ่ม (CKSize : Chunk Size) มีหน่วยเป็นไบต์ (Byte) โดยจะไม่รวมไบต์ที่ใช้ไปในการเก็บชื่อกลุ่ม (CKID) แต่จะเก็บตัวเลขบอกขนาดของกลุ่มเองด้วย ตัวเลขบอกขนาดของกลุ่มจะแสดงด้วยข้อมูล 4 ไบต์โดยไบต์ที่มีความสำคัญต่ำสุด (Least Significant Byte) จะถูกเก็บเข้าไปก่อน

3. ข้อมูลของกลุ่ม (CKData : Chunk Data) มีจำนวนไบต์เท่ากับตัวเลขที่แสดงในข้อที่ 2 เช่น ในกลุ่มรูปแบบของข้อมูล (Format Data Chunk) จะมีข้อมูลของกลุ่มเป็นรูปแบบในการบันทึกเสียง ยกตัวอย่างเช่น อัตราแซมปลิ้ง , จำนวนบิทต่อ 1 แซมเปิล เป็นต้น หรือในกลุ่มข้อมูลจากการแซมปลิ้ง (Sample Data Chunk) จะมีข้อมูลของกลุ่มเป็นแอมปลิจูด (Amplitude) ที่ได้จากการแซมปลิ้ง

POSITION HEX	DEC		nBYTE
0000	0	CKID	4
0004	4	CKSize	4
0008	8	FCC TYPE	4
000C	12	CKID	4
0010	16	CKSize	4
0014	20		2
0016	22		2
0018	24		4
001C	28		4
0020	32		2
0022	34		2
0024	36	CKID	4
0028	40	CKSize	4
002C	44		



รูปที่ 2.6 แสดงโครงสร้างเวฟไฟล์ที่มีรูปแบบของไฟล์ RIFF

รูปแบบการบันทึก	ค่าสูงสุด	ค่าต่ำสุด	ค่าที่ตำแหน่งตรงกลาง
8 บิท พีซีเอ็ม	255 (FFH)	0	128 (80H)
16 บิท พีซีเอ็ม	32767 (7FFFH)	-32768 (-800H)	0

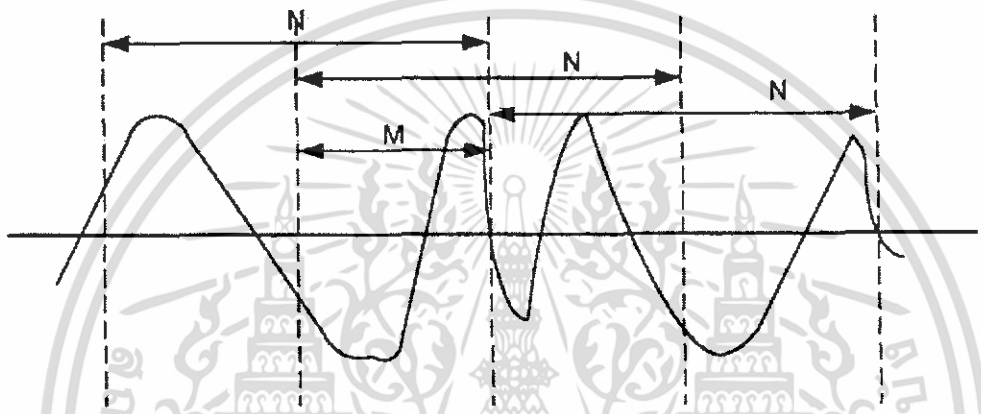
รูปที่ 2.7 แสดงค่าสูงสุด ค่าต่ำสุด ค่ากลางของรูปแบบการบันทึกแต่ละอัน

2.8 การแบ่งช่วงสัญญาณ (Frame Blocking)

ข้อมูลเสียงซึ่งอยู่ในโดเมนเวลา จะถูกตัดมาวิเคราะห์ทีละเฟรม เฟรมละ N จุด โดยมีจุดเหลื่อมกัน M จุด จนหมดสัญญาณเสียงที่นำมาวิเคราะห์ การเลือกใช้ค่า N และ M พิจารณาได้ดังนี้

ก) ค่า N เล็กไป จะทำให้ได้สเปกตรัมที่ไม่ละเอียด (Low Resolution) แต่ถ้าค่า N ใหญ่ไป ก็จะลดความสามารถในการติดตามการเปลี่ยนแปลงของสเปกตรัมลง จึงไม่เหมาะสมกับสัญญาณที่มีสเปกตรัมเปลี่ยนแปลงรวดเร็ว

ข) M เล็กไป จะลดความสามารถในการติดตามการเปลี่ยนแปลงของสเปกตรัมลง แต่ถ้า M ใหญ่ไป ก็จะสิ้นเปลืองการประมวลผลมาก



รูปที่ 2.8 แสดงการแบ่งช่วงสัญญาณ

2.9 การวินโดว์ (Windowing)

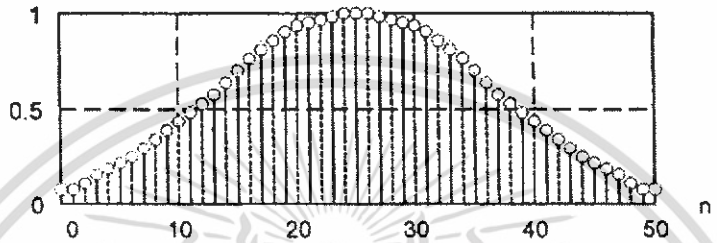
เนื่องจากเราใช้วิธีตัดสัญญาณเพื่อมาหาสเปกตรัมเป็นบล็อก ๆ ซึ่งเท่ากับเป็นการหาสเปกตรัมของ "บล็อกของสัญญาณ" ไม่ใช่ตัวสัญญาณจริง ๆ ที่เข้ามาติดต่อกัน ไม่มีจุดเริ่มต้นหรือสิ้นสุด การตัดสัญญาณเป็นบล็อกนี้จะทำให้เกิดความคลาดเคลื่อนในสเปกตรัมที่ได้ วิธีลดผลของความคลาดเคลื่อนนี้ทำได้โดยคูณสัญญาณแต่ละบล็อก หรือ $X_l(n)$ ด้วยฟังก์ชั่นหน้าต่าง (window function) ก่อนที่จะทำการหาสเปกตรัม

ฟังก์ชั่นหน้าต่างมีหลายแบบ เป็นฟังก์ชั่นที่มีลักษณะสมมาตร และมีค่าสูงสุดที่จุดกึ่งกลาง มีความยาวเท่าไหร่ก็ได้ตามต้องการ (ตามค่าที่แทนลงในสูตร) ตัวอย่างของฟังก์ชั่นหน้าต่างที่เป็นที่นิยมก็คือ หน้าต่างแฮมมิ่ง (Hamming window) ซึ่งมีสมการคือ

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad ; n = 0,1,2,\ldots,N-1 \qquad (2.1)$$

$$x_{mew}(n) = x_t(n)w(n) \qquad (2.2)$$

เราต้องการ w(n) ที่มีความยาวเท่ากับ 1 บล็อกสัญญาณ ดังนั้นต้องใช้ N เท่ากับความยาว 1 บล็อกสัญญาณ รูปร่างของหน้าต่างแบบแฮมมิ่งแสดงดังรูปที่ 2.9 ซึ่งจะสังเกตเห็นได้ว่า ฟังก์ชั่นหน้าต่าง
มีค่าเล็กที่ส่วนต้นและส่วนปลาย ซึ่งเป็นส่วนที่ลดผลของการเปลี่ยนแปลงที่จุดเริ่มต้น และจุดสิ้นสุดของบล็อก



รูปที่ 2.9 ตัวอย่างของฟังก์ชั่นหน้าต่างแบบแฮมมิ่ง ที่ N = 50

2.10 การวิเคราะห์ฟูริเยร์ (Fourier Analysis)

การวิเคราะห์ฟูริเยร์ช่วงสั้นเป็นเทคนิคการหาความถี่ที่ใช้กันมานานแล้ว การวิเคราะห์ฟูริเยร์ให้ตัวแทนของสัญญาณเสียงพูดเป็นฟังก์ชั่นความถี่ในเทอมของขนาดและเฟส เนื่องจากเสียงพูดไม่ได้นิ่งตลอดเวลา ดังนั้นจึงจำเป็นต้องทำการวิเคราะห์ในช่วงสั้น ๆ โดยใช้วินโดว์

การแปลงฟูริเยร์ช่วงสั้นมีสมการดังนี้

$$X_n(e^{-jw}) = \sum_{m=-\infty}^{\infty} x(m)[\exp(-jwm)w(n-m)] \qquad (2.3)$$

ในการคำนวณต้องใช้ DFT แทนการแปลงฟูริเยร์แบบต่อเนื่อง โดยใช้ฟังก์ชั่นวินโดว์ลดข้อมูลแบบไม่ต่อเนื่องทั้งหมดให้เหลือจำนวน N ตัว (N คือช่วงเวลาหรือขนาดของวินโดว์ที่ใช้ในการแปลง DFT) ข่าวสารต่าง ๆ ใน $X_n(e^{jw})$ จะไม่สูญหายไปจากข้อมูลเดิมถ้าการแปลงนั้นสุ่มมาด้วยความถี่สูงเพียงพอ (คือช่วงระยะห่างระหว่าง N) และวินโดว์ w(n) ไม่มีจุดสุ่มที่เป็นศูนย์ตลอดช่วง N ตัวแปร N เป็นตัวแปรที่จะต้องระวังมากเป็นพิเศษในการวิเคราะห์ความถี่ช่วงสั้น ถ้าค่าของ N ต่ำจะทำให้ความละเอียดในโดเมนความถี่หยาบมาก เพราะจะให้ผลที่ดีในโดเมนเวลา เพราะการเฉลี่ยถูกทำในช่วงสั้น ๆ เท่านั้น ในทางตรงกันข้ามถ้า N มีขนาดใหญ่จะให้ผลของความละเอียดเวลาที่แย่ในโดเมนเวลา แต่จะทำให้โดเมนความถี่มีความละเอียดสูงกว่า

2.10.1 การแปลงฟาสท์ฟูริเยร์ (Fast Fourier Transform)

โดยทั่วไปการแปลงฟูริเยร์ (Discrete Fourier Transform) เป็นการคำนวณที่ใช้เวลาค่อนข้างมาก ลำดับขั้นตอนในการคำนวณ DFT ให้เร็วขึ้นเรียกว่า ฟาสท์ฟูริเยร์ (Fast Fourier Transform หรือ FFT) โดยการแปลง FFT แบ่งได้เป็น 2 ชนิดใหญ่ ๆ คือ ชนิดลดทอนทางเวลา (Decimation In Time หรือ DIT) และชนิดลดทอนทางความถี่ (Decimation In Frequency หรือ DIF) สำหรับในส่วนนี้จะแสดงเฉพาะชนิดลดทอนทางเวลาซึ่งเกี่ยวกับในโครงงานนี้เท่านั้น

2.10.2 ขั้นตอนวิธีลดทอนทางเวลา

วิธีนี้เป็นการจัดแบ่งกลุ่มลำดับสัญญาณในโดเมนเวลา X(n) ขนาด N จุดออกเป็น 2 ลำดับสัญญาณขนาด N/2 จุดเท่ากันคือ ลำดับคู่และลำดับคี่ โดยที่ลำดับคู่เกิดจากการเอาลำดับในตำแหน่งคู่มาเรียงกัน ที่เหลือเป็นลำดับคี่ ดังนั้นจะได้

$$x_e(m) = x(2n) \qquad ; m = 0,1,2,...,(N/2)-1$$
$$x_o(m) = x(2n+1) \qquad ; m = 0,1,2,...,(N/2)-1 \qquad (2.4)$$

ถ้าให้ W_N เท่ากับ exp(-2j/N) จะทำให้การคำนวณ DFT ของลำดับ x(n) ที่ยาว N จุด สามารถเขียนใหม่ได้เป็น

$$X(k) = \sum_{m=0}^{N/2-1} x_e(2m)W_k^{2km} + \sum_{m=0}^{N/2-1} x(2m+1)W_N^{(2m+1)k} \qquad (2.5)$$

โดยที่

$$W_N^2 = \{\exp[j2\pi / N]^2\} = \exp[j2\pi / N/2] = W_{N/2} \qquad (2.6)$$

ซึ่ง $W_{N/2}$ เป็นค่า W ลำดับความยาว N/2 จุด สมการ 2.5 สามารถเขียนใหม่ได้เป็น

$$X(k) = \sum_{m=0}^{N/2-1} x_e(m)W_{N/2}^{km} + \sum_{m=0}^{N/2-1} x(m)W_{N/2}^{km} \qquad (2.7)$$

การนำผลการแปลง DFT ขนาด 2 จุด จำนวน N/2 ภาคมารวมกัน เพื่อให้เป็นการคำนวณ DFT ขนาด N จุด จะต้องมีหลักเกณฑ์ที่ถูกต้องด้วย จากสมการ 2.7 ถ้าเขียนให้อยู่ช่วง $0 \leq k \leq N/2 - 1$ สามารถเขียนใหม่ได้เป็น

$$X(k) = X_e(k) + (W_N^k)X_0(k) \qquad ;0 \le k \le N/2-1$$

$$= X_e(k-N/2) + (W_N^k)X_0(k-N/2) \quad ;0 \le k \le N/2-1 \qquad (2.8)$$

เทอม W_N^k เรียกว่า ตัวประกอบหมุน (Twiddle Factor) ซึ่งใช้ร่วมกันกับ DFT ขนาด 2 จุด หรือขนาด N/2 จุด ในการนำมาประกอบเป็น DFT ขนาด N จุดได้เหมือนเดิม และจากความสัมพันธ์

$$(W)_N^{k-N/2} = -(W)_N^k$$

จะได้

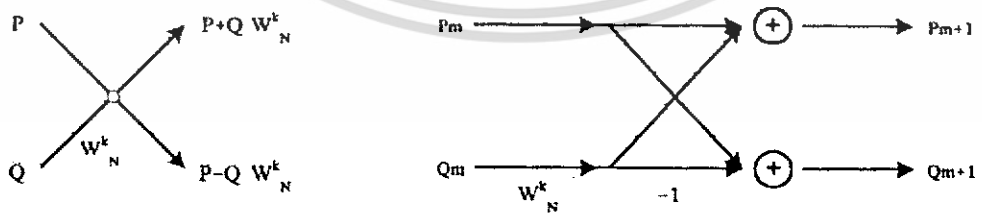
$$X(k) = X_1(k) + (W)_N^k X_2(k) \qquad ;0 \le k \le N/2-1$$

$$= X_1(k-N/2) + (W)_N^{k-N/2} X_2(k-N/2) \qquad ;N/2 \le k \le N/2-1 \qquad (2.9)$$

ผลจากสมการนี้สามารถนำไปใช้สร้างหน่วยคำนวณที่เรียกว่า หน่วยผีเสื้อ (Butterfly Unit) โดยมีข้อมูลเข้าคือ A และ B และข้อมูลออก คือ X และ Y เป็น

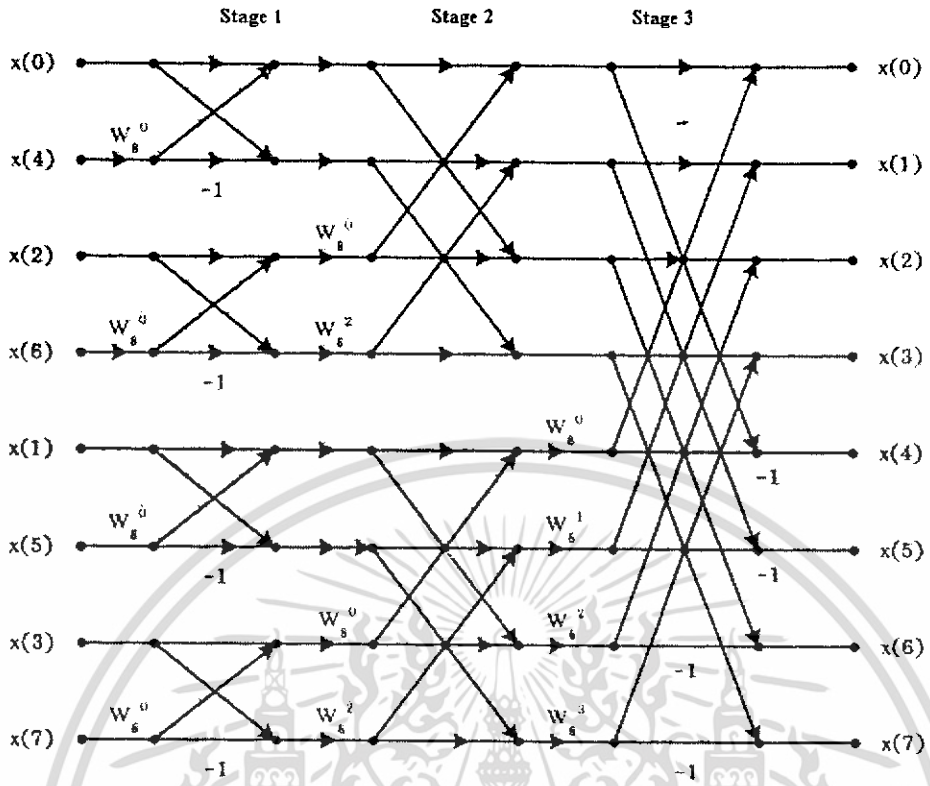
$$X = A + (W)_N^k B$$
$$Y = A + (W)_N^k B \qquad (2.10)$$

ซึ่งสามารถเขียนอธิบายแทนด้วยโฟล์วชาร์ทดังแสดงในรูป 2.10



รูปที่ 2.10 แสดงหน่วยผีเสื้อของการคำนวณตามขั้นตอนวิธีฉดทอนทางเวลา

สำหรับตัวอย่างการคำนวณ DFT โดยใช้ FFT แสดงดังรูป 2.11



รูปที่ 2.11 แสดงวิธีการของ FFT แบบลดทอนทางเวลา (DIT) สำหรับข้อมูลขนาดจุด

2.11 ความรู้เบื้องต้นเกี่ยวกับนิวรัลเน็ตเวอร์ค

ในช่วงระยะเวลา 8-9 ปีที่ผ่านมา ในต่างประเทศมีการตื่นตัวในการวิจัยและพัฒนาเกี่ยวกับโครงข่ายประสาทเทียม (Artificial Neural Networks : ANN) อย่างกว้างขวางทั้งทางด้านทฤษฎีและการประยุกต์ใช้งาน จนกระทั่งถึงปัจจุบันได้มีการประยุกต์นำนิวรัลเน็ตเวอร์คมาใช้ในอุปกรณ์เครื่องใช้ต่างๆมากขึ้น เช่น เครื่องมือหาปลา (Sonar) ที่มีความฉลาดมากขึ้น เช่น สามารถบอกได้ว่าฝูงปลาที่กำลังตรวจจับอยู่นั้นเป็นปลาชนิดใด จำนวนเท่าไร เครื่องโทรศัพท์แบบที่สามารถเรียกเลขหมายปลายทางให้อัตโนมัติ (Voice Phone) เพียงยกหูแล้วพูดชื่อของผู้ที่จะติดต่อเท่านั้น เครื่องอ่านตัวอักษร (OCR) ที่สามารถเปลี่ยนภาพตัวอักษรให้เป็นรหัสตัวอักษรแบบแอสกี้ ระบบนักบินอัตโนมัติ (Auto Pilot Aircraft) ระบบการคาดเดาอนาคตจากข้อมูลในอดีต (Forcasting, Prediction) ฯลฯ ซึ่งเครื่องมือและอุปกรณ์ต่างๆที่นำเอา ANN มาใช้ช่วยวิเคราะห์นั้น จะมีความฉลาดมากขึ้นและมีระบบความคิดที่มีการทำงานในลักษณะคล้ายกับมนุษย์

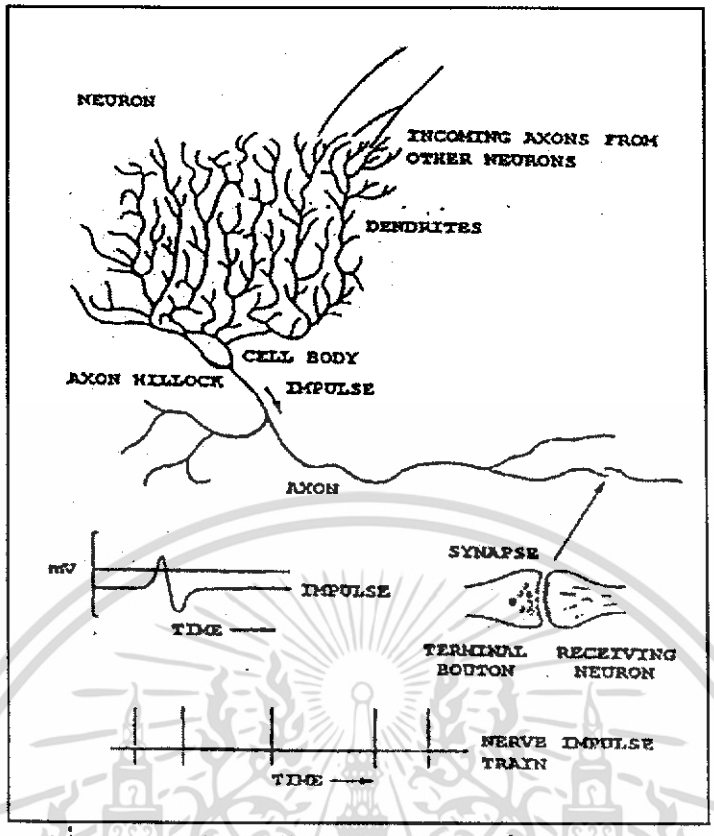
นิวรัลเน็ตเวอร์ค หมายถึง โครงข่ายใยประสาทที่เชื่อมต่อกันระหว่างเซลล์ประสาทจำนวนมากมายมหาศาล มีความสามารถประมวลผลสูงบรรจุอยู่ในสมอง สมองชีวภาพที่เป็นจุดศูนย์กลางการควบคุมกิจกรรมของการดำเนินชีวิตการวิจัยสร้างโครงข่าประสาทเทียม (Artificial Neural Network) มีแนวคิดเลียนแบบการทำงานของสมองชีวภาพ โดยเรียนรู้และศึกษาการทำงานของสมอง

ชีวภาพเพื่อกำหนดแนวทางสำหรับการสร้างแบบจำลองขึ้นมา แล้วพยายามสมมติฐานลักษณะการทำงานโดยจำลองเป็นโมเดลคณิตศาสตร์ที่มีลักษณะเดียวกันแล้วดำเนินการคำนวณโดยใช้คอมพิวเตอร์

2.11.1 นิวรัลเน็ตเวอร์คชีวภาพ

ระบบคิดคำนึงของมนุษย์ มีโครงสร้างพื้นฐานจากเซลล์สมองที่เรียกว่านิวรัล (Neural) เรียงเป็นชั้นๆ อย่างซับซ้อนจำนวนมหาศาลมหาศาลประมาณหมื่นล้าน (10^{11}) นิวรัล และอาจมีจุดเชื่อมโยงภายในถึงพันล้านล้าน (10^{15}) จุด แต่ละนิวรัลจะมีคุณลักษณะแตกต่างกันไปโดยมีการทำงานคล้ายกัน คือ รับเข้า, ประมวลผล, ส่งออกสัญญาณไฟฟ้าเคมีผ่านไปยังนิวรัล ซึ่งจะส่งสื่อสารไปตามระบบของสมอง

จากภาพที่ 2.12 ส่วนแขนงที่ขยายแยกออกไปจากตัวเซลล์ต่อไปยังเซลล์อื่นๆเพื่อรับสัญญาณเรียกว่า เดนไดรท์ (Dendrites) จุดรับสัญญาณจากเซลล์อื่น เข้ามายังตัวเซลล์ จะผ่านมาทางจุดเชื่อมต่อที่เรียกว่า ซินแนปส์ (Synapse) ซึ่งแอกซอน (Axon) จะเป็นตัวส่งสัญญาณเอาท์พุท ออกไปยังนิวรอลอื่น จากผลการวิจัยพบว่า แต่ละนิวรอลจะเชื่อมต่อออกไปยังนิวรอลอื่นๆ ซึ่งแต่ละนิวรอลจะมีคุณสมบัติในการเพิ่ม ขยาย หรือลดทอนความเข้มของสัญญาณบางสัญญาณที่เข้ามาทางเดนไดรท์ของเซลล์ (ซึ่งมีแขนงมากมาย) อาจสามารถกระตุ้นตัวเซลล์ แต่บางสัญญาณก็อาจจะยับยั้งตัวเซลล์ เนื่องจากเซลล์ประสาทหนึ่งเซลล์ มีเดนไดรท์มาก ฉะนั้น สัญญาณกระตุ้นจากเดนไดรท์ ที่รับเข้ามาจากเซลล์ประสาทอื่นๆ จะถูกนำมารวมกันที่ตัวเซลล์ประสาทที่เซลล์ประสาทจะมีค่าเทรสโฮล (Threshold) ค่าหนึ่งหากผลรวมของเซลล์ไฟฟ้าเคมี (Electrochemical) มีค่ามากกว่า เทรสโฮลเซลล์เซลล์ประสาทก็จะส่งสัญญาณขนาดหนึ่งผ่านทางแอกซอน ไปยังนิวรอลอื่นๆ การจัดเรียงชั้น (Layer) และลักษณะการเชื่อมโยงระหว่างนิวรอลในสมองนั้น มีการจัดเรียงที่ซับซอน สอดคล้องกับหน้าที่การทำงานเฉพาะส่วน มีการเจริญเติบโตสัมพันธ์กับสิ่งแวดล้อม และมีการเรียนรู้ตลอดเวลา ซึ่งใช้เวลานานนับปี ดังนั้นจึงยากที่จะสร้างโมดุลขึ้นมา เพื่อเลียนแบบให้มีคุณลักษณะคล้ายสมองชีวภาพได้ทั้งหมด ผลงานที่ได้จากการทำวิจัยในปัจจุบันเป็นเพียงการจำลอง การเลียนแบบ และการทำงานเฉพาะบางส่วนของโครงข่ายประสาทมาใช้เฉพาะกับงานใดงานหนึ่ง ซึ่งมีการวิจัยลักษณะของโครงข่ายแบบต่างๆขึ้นมา โดยแต่ละแบบจะเหมาะกับงานประเภทหนึ่งๆเท่านั้น

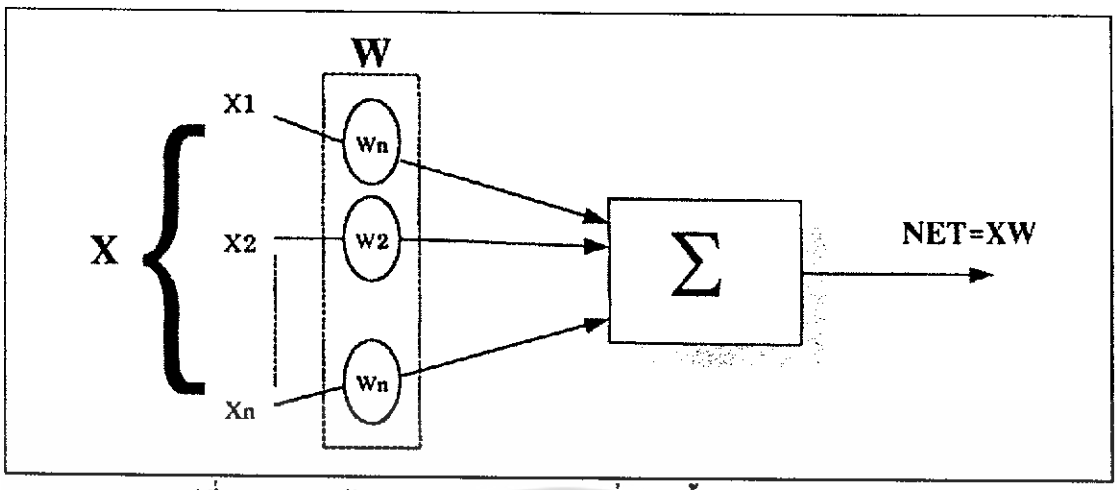


รูปที่ 2.12 แสดงโครงสร้างตัวอย่างของเซลล์ประสาทชีวภาพ

2.11.2 โครงข่ายประสาทเทียม (Artificial Neural Networks)

การออกแบบสร้างประสาทเทียมนั้น มีสมมติฐานขั้นแรกจากคุณสมบัติ ของระบบประสาทชีวภาพ ดังที่กล่าวมา กล่าวคือ ชุดรับสัญญาณข้อมูล อินพุทของเซลล์ประสาทหนึ่ง ได้จากสัญญาณเอาท์พุทของเซลล์ประสาทอื่นๆ ผ่านทางซินแนปส์และเดนไดรท์ ข้อมูลแต่ละค่าที่รับมาจะถูกลดขนาดด้วย ซินเนปติกส์ ซึ่งภายในประกอบด้วยสารเคมีประเภท K^+, Ca^{++}, Na^+, Cl^- ซึ่งจะมีลักษณะทางความนำพัลส์ (Pulse) สัญญาณไฟฟ้าเคมีที่แตกต่างกัน (James A. Freeman and David M.Skapura, 1991:8-9) ด้วยเหตุนี้ โมดุลประสาทเทียมที่สร้างขึ้น จะต้องมีการถ่วงน้ำหนักให้กับโมดุลก่อนที่จะนำเข้าสู่โมดุลประสาทเทียม จุดนี้เรียกว่า น้ำหนักซินแนปติกส์ ปริมาณของข้อมูลที่เข้าสู่นิวรอล จะถูกนำมารวมกัน และตัดสินใจด้วยระดับความสนใจของนิวรอล (Activation level) แล้วจะส่งเป็นเอาท์พุทออกที่แอกซอนไปยังนิวรอลอื่นๆ

จากภาพที่ 2.13 แสดงถึงโมดุลที่สร้างขึ้นโดยแนวความคิดจากเซลล์สมองชีวภาพ สัญญาณอินพุท คือ $X_1, X_2, \ldots\ldots, X_n$ จะถูกป้อนเข้าไปยังนิวรอลที่สร้างขึ้น ซึ่งเปรียบเทียบได้กับสัญญาณที่ป้อนเข้ายังซินแนปส์ของนิวรอลชีวภาพ สัญญาณอินพุทนี้จะนำไปคูณกับค่าน้ำหนักแนปติกส์ที่มีค่าตั้งแต่ 0.00-1 (Weight : ค่าที่ใช้ถ่วงน้ำหนัก) $W_1, W_2, \ldots\ldots, W_n$ ก่อนที่จะเข้าสู่ล็อกซัมเมชั่น (Σ : Summation) ซึ่งค่าถ่วงน้ำหนักนี้ จะสอดคล้องกับค่าสเตรงท์ (Strength) ของจุดต่อซินแนปส์ชีวภาพ



รูปที่ 2.13 แสดงไดอะแกรมของนิวรอลที่สร้างขึ้น (Artificail Neuion)

แต่ละจุด (Single biological synaptic connection) บล็อกซัมเมชั่นนี้ ก็จะทำหน้าที่สอดคล้องคล้ายกับตัวเซลล์สมองชีวภาพ ผลรวมทางคณิตศาสตร์ของอินพุท และน้ำหนักจะได้เป็นเอาท์พุท เราเรียกว่า เน็ต (NET) ซึ่งเราจะรวมกันในรูปของเวกเตอร์ได้ดังนี้

$$NET = X_1W_1 + X_2W_2 + \ldots\ldots\ldots + X_nW_n \qquad (2.11)$$

จะได้

$$NET = XW \qquad (2.12)$$

2.11.3 ฟังก์ชั่นกระตุ้นความสนใจ (Activation Function)

เมื่อได้สัญญาณ NET แล้ว กระบวนการต่อมาที่นิวรอลต้องทำคือตัดสินใจ เราจึงต้องกำหนดฟังก์ชั่นการตัดสินใจ เพื่อใช้เป็นระดับของการตัดสินใจให้กับนิวรอล เพื่อให้ได้สัญญาณเอาท์พุทของนิวรอลออกมา ซึ่งเชื่อมต่อไปยังนิวรอลตัวอื่นๆเป็นโครงข่าย OUT ที่ได้จากอาจเป็น Simple linear function โดย

$$OUT = K[NET] \qquad (2.13)$$

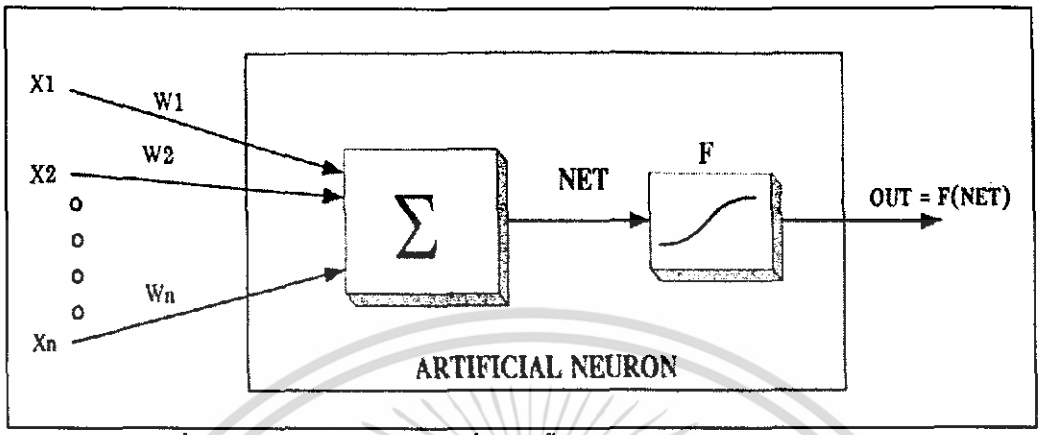
โดย K เป็นค่าคงที่ ที่เรียกว่า Threshold function

ตัวอย่างเช่น

OUT = 1 ถ้า NET>T

OUT = 0 เมื่อเป็นกรณีอื่นๆ (2.14)

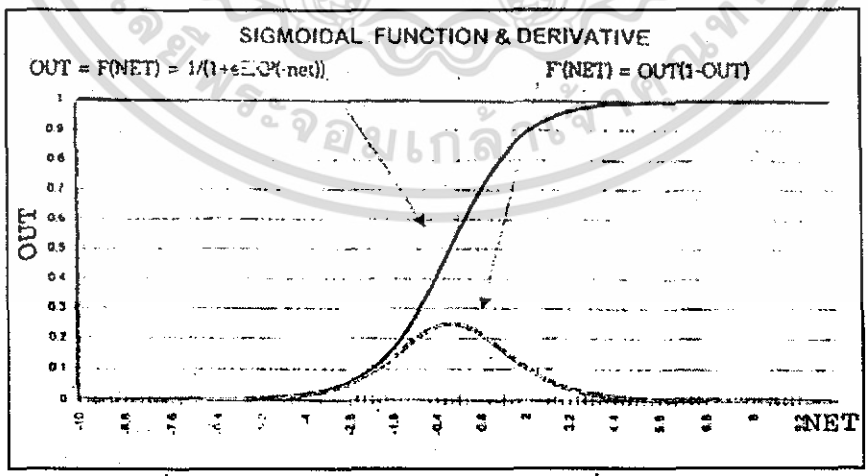
และ T เป็นค่าเทรสโฮลดงที่ หรืออาจเป็น Function อื่นๆ ที่เลียนแบบคุณสมบัติไม่เป็นเชิงเส้นของเซลล์ประสาทชีวภาพได้อย่างใกล้เคียงกว่า และใช้เป็นฟังก์ชั่นให้กับโครงข่ายทั่วไปได้



รูปที่ 2.14 แสดงโมดุลนิวรอลที่สร้างขึ้นร่วมกับ Activation Function

ในภาพที่ 2.14 บล็อก F จะได้รับผลที่ได้จาก NET มาสร้างเป็นสัญญาณเอาท์พุทที่ OUT โดยกระบวนการภายในบล็อก F จะบีบช่วงของ OUT ให้อยู่ในขอบเขตจำกัด ตามต้องการ ดังนั้น ค่า OUT จะมีค่าไม่ต่ำกว่าช่วงที่กำหนดโดยค่าของ NET เราเรียกบล็อก F นี้ว่าสแควชชิ่งฟังก์ชั่น (Squashing function) และโดยทั่วไป สแควชชิ่งฟังก์ชั่นที่ใช้เป็นแบบลอจิสติกฟังก์ชั่น หรือซิกมอยด์ (Logistic function or "Sigmoid") ซึ่งมีรูปร่างคล้ายตัว S โดยเขียนเป็นสมการคณิตศาสตร์ได้ดังนี้คือ

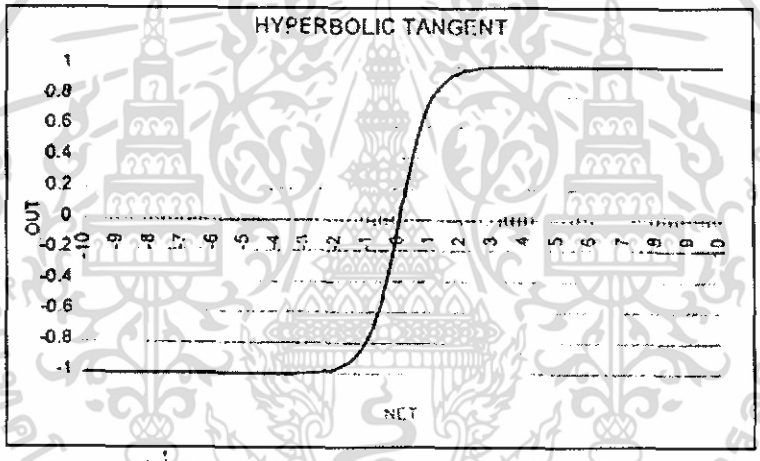
$$F(x) = \frac{1}{(1-e^{-x})} \tag{2.15}$$



รูปที่ 2.15 แสดงกราฟที่ได้จากสมการซิมมอยด์ลอจิสติกฟังก์ชั่น (Sigmoidal logistic function)

ลักษณะของเทรสโฮลฟังก์ชั่นมีลักษณะเป็น Non-linear function เช่น S-Curve เราจะได้ค่าเอาท์พุท ที่มีความไวต่อสัญญาณอินพุทที่มีขนาดเล็กๆ และเฉลี่ยต่อสัญญาณแรงๆ ซึ่งสัญญาณอ่อนๆ ไปทางบอกเพียงเล็กน้อยก็จะทำให้ OUT ใกล้เคียง "1" กระตุ้นหรือสัญญาณอ่อนๆทางลบเพียงเล็กน้อย ก็จะทำให้ Output ใกล้เคียง "0" (ยับยั้ง) ขณะที่สัญญาณแรงๆทางบอกก็ยังคงให้ Output ใกล้เคียง "1" และสัญญาณทางลบแรงๆ ก็คงให้ Output ใกล้เคียง "0" เช่น คุณลักษณะแบบนี้ เป็นแบบ NON-LINEAR GAIN ซึ่งคลอสส์เบอร์ก (Grossberg, 1973) พบว่า คุณลักษณะที่เป็น Non-linear gain นี้สามารถแก้ปัญหา Noise-saturation dilemma ได้ และทำให้นิวรอลเทียมที่สร้างขึ้นสามารถทำงานกับขนาดของอินพุทได้กว้างมากขึ้น

ยังมีฟังก์ชั่นอื่นๆอีกคือ ไฮเปอร์โบลิก แทนเจนท์ (Hyperbolic tangent) มันจะมีลักษณะคล้ายกับ Logistic function และนิยมใช้บ่อยๆ ในการสร้างโมดุลคณิตศาสตร์ การกระตุ้นเร้าความสนใจของเซลล์สมองเทียม ซึ่งมีคุณสมบัติคล้ายชีวภาพของเซลล์สมองคือ OUT = Tanh(X)



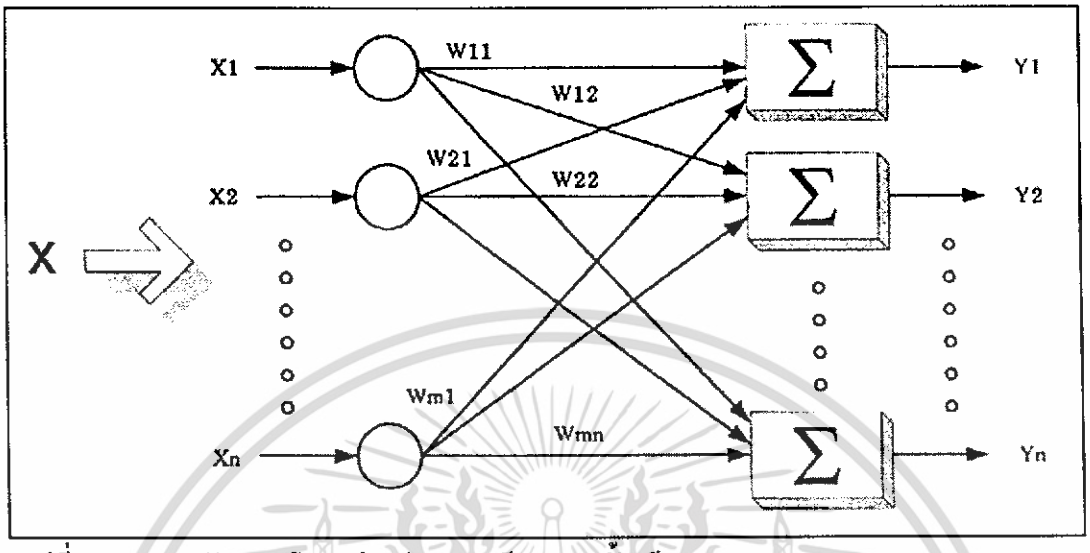
รูปที่ 2.16 แสดง Hyperbolic Tagent Function

จากภาพที่ 2.16 ส่วนที่เหมือนกับ ซิกมอยด์ ลอจิสติก ฟังก์ชั่น คือมีลักษณะเป็น S แต่เนื่องจากมันจะมีความสมมาตร จึงให้ OUTPUT อยู่ระหว่าง "-1" ถึง "1" OUTPUT จะเป็น "0" เมื่อ NET เป็น "0" OUTPUT เข้าใกล้ "1" เมื่ออินพุทไปทางบวก และเข้าใกล้ "-1" เมื่ออินพุทไปทางบวก และเข้าใกล้ "-1" เมื่ออินพุทมีทิศทางไปทางลบ

2.11.4 โครงข่ายประสาทเทียมแบบชั้นเดียว (Single Layer Artificial Neural Networks)

ที่กล่าวมาจนถึงจุดนี้ เป็นการกล่าวถึงหลักการและเหตุผลในการสร้างเซลล์ประสาทเทียมเพียงหนึ่งเซลล์ โดยใช้แนวความคิดจากเซลล์ประสาทชีวภาพ การจะนำเซลล์ประสาทเทียมมาใช้งานได้นั้น ต้องใช้เซลล์ประสาทเทียมที่มีคุณลักษณะต่างๆกัน (ค่า Weight จะทำให้คุณสมบัติของเซลล์ประสาทเทียม แต่ละเซลล์มีคุณสมบัติแตกต่างกันออกไป) มาเชื่อมโยงเป็นโครงข่ายในลักษณะ

เดียวกับเซลล์สมองชีวภาพเสียก่อน ซึ่งลักษณะการเชื่อมโยงมีหลายชนิด แต่ละชนิดก็มีคุณลักษณะเด่นที่แตกต่างกันไป



รูปที่ 2.17 แสดงลักษณะโครงข่ายประสาทเทียมแบบชั้นเดียว (Single-Layer Neural Network)

จากรูป 2.17 เป็นโครงข่ายประสาทเทียมแบบชั้นเดียว ที่ประกอบด้วยเซลล์ประสาทเทียมง่ายๆหลายๆชุด ความสามารถในการคำนวณของโครงข่ายประสาทเทียม ได้มาจากลักษณะการเชื่อมต่อ เป็นโครงข่ายประสาทเทียมโครงข่ายง่ายๆ เป็นกลุ่มโมดุลประสาทเทียมที่เชื่อมต่อกันเป็นชั้นๆ (Layer) ในภาพที่ 2.18 เป็นโครงข่ายประสาทเทียมแบบชั้นเดียว (Single layer) ที่ประกอบด้วยเอาท์พุทเลเยอร์ (กลุ่มของบล็อกซัมเมชั่นที่อยู่ทางขวามือ) และอินพุทเลเยอร์ (วงกลมทางซ้ายมือ) โดยไม่พิจารณาอินพุทเลเยอร์ว่าเป็นนิวรอลเลเยอร์ เนื่องจากอินพุทเลเยอร์จะทำหน้าที่เชื่อมต่ออินพุทที่รับมา และส่งออกไปให้ยังแต่ละอินพุทนิวรอลเลเยอร์ (ในที่นี้คือ Output layer) ในชั้นถัดไป โดยแต่ละอินพุทจะถูกคูณ โดยค่าน้ำหนักเฉพาะแต่ละอินพุท โครงข่ายประสาทเทียมที่สร้างขึ้นในขั้นแรกไม่ซับซ้อน โดยแต่ละนิวรอลจะได้เอาท์พุทจาก

OUT = Logistic Function คูณ (ผลรวมของ Input X กับ Weight)

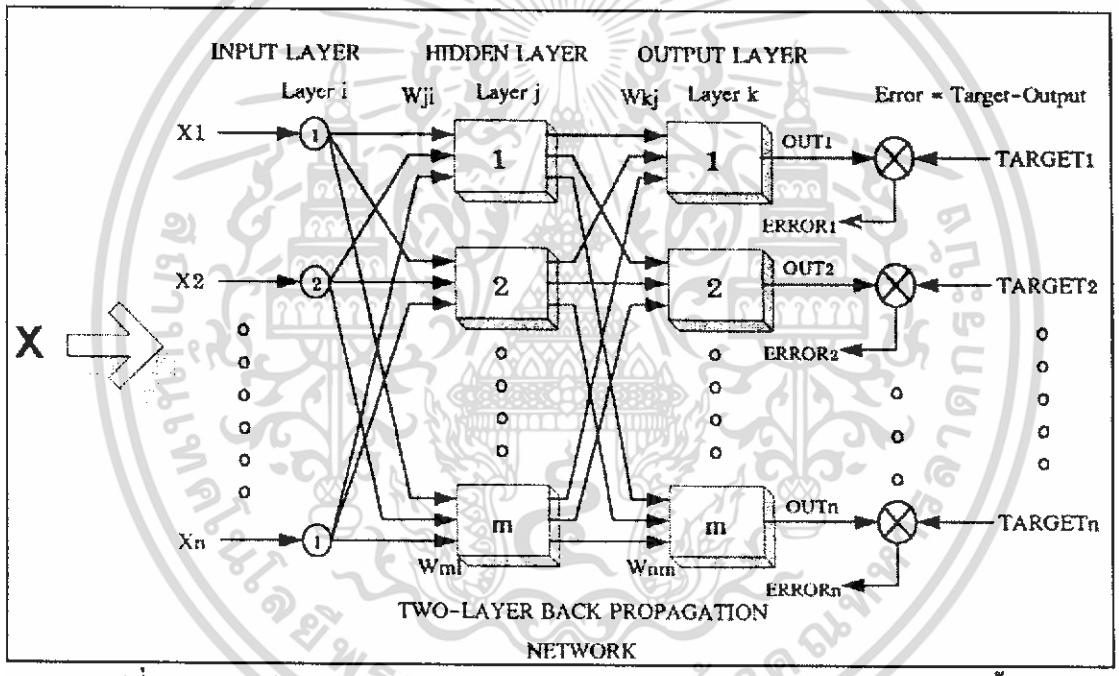
หรือ OUT = F(NET) (2.16)

อย่างไรก็ดีลักษณะการเชื่อมโยงระหว่างโครงข่ายไม่ได้มีแบบเดียว การเชื่อมโยงระหว่างเลเยอร์อาจมีการเชื่อมโยงย้อนกลับมาที่อินพุทเลเยอร์อีก ซึ่งโรงข่ายประสาทชีวภาพก็มีลักษณะดังกล่าวเช่นกัน สำหรับค่า Weight ในภาพที่ 2.18 มีวิธีการพิจารณาในรูปของ เมตริกน้ำหนัก (Weight matrix) ซึ่งหากโครงข่ายมีหลายชั้น จะช่วยให้ระบุค่าน้ำหนักได้ง่ายขึ้น และเพื่อหลีกเลี่ยงความสับสนจะกำหนดเป็นไดเมนชั่น (Dimensions) ของเมตริก โดยให้ m แทนจำนวนแถว หรือจำนวนของอินพุท

และ n แทนจำนวนของนิวรอล ที่สร้างขึ้น ตัวอย่างเช่น น้ำหนักที่เชื่อมระหว่างอินพุทตัวที่ 4 กับนิวรอลตัวที่ 2 คือ $W_{4,2}$

2.11.5 โครงข่ายประสาทเทียมแบบหลายชั้น (Multilayer Artificial Neural Nettworks)

โครงข่ายที่ซับซ้อนจะมีความสามารถในการคำนวณที่ดีขึ้นมันจะเป็นโครงข่ายที่มีโรงสร้างเป็นจินตนาการที่น่าเป็นไปได้โดยการจัดการเชื่อมโยงนิวรอล มีโครงสร้างเป็นชั้นๆ คล้ายส่วนหนึ่งของสมอง และมีการพัฒนาอัลกอริทึมเกี่ยวกับการฝึกสอนให้โรงข่ายแบบหลายชั้นทำงานได้ ตามความต้องการแล้วเมื่อไม่นานมานี้ โรงข่ายแบบหลายชั้นอาจจะสร้างจาก กลุ่มของโครงข่ายแบบชั้นเดียวเอาท์พุทของ Layer หนึ่ง จะใช้เป็นอิทพุทของ Layer ถัดไป ในภาพที่ 2.18 แสดงเน็ตเวอร์คที่มีการเชื่อมต่อแบบสองชั้น



รูปที่ 2.18 แสดงไดอะแกรมของ Backpropagation Neural Networks แบบสองชั้น

ภาพที่ 2.18 โครงข่ายประสาทเทียมแบบหลายชั้น ที่ต่อเชื่อมโยงแบบเต็มขั้น ในโครงข่ายแบบหลายชั้นมีการเรียกชื่อชั้นต่างๆ ดังนี้ คือ ชั้นที่ต่อโดยตรงกับอินพุท เรียกว่า อินพุทเลเยอร์ (Input layer) ชั้นนี้จะไม่มีการคำนวณ แต่จะทำหน้าที่ต่อเชื่อมข้อมูลไปยังชั้นถัดไป ชั้นที่อยู่ท้ายสุดทางขวามือเรียกว่า เอาท์พุทเลเยอร์ (Output layer) เป็นชั้นที่โครงข่ายจะให้ผลลัพธ์ ส่วนชั้นที่อยู่ระหว่างอินพุทเลเยอร์ และเอาท์พุทเลเยอร์ จะมีกี่ชั้นก็ตามจะเรียกว่า ฮิดเดนเลเยอร์ (Hidden layer) หากฮิดเดนเลเยอร์ มีหลายๆชั้น ก็จะมีการตั้งชื่อเฉพาะลงไปให้กับแต่ละชั้น

2.11.6 ฟังก์ชั่นกระตุ้นความสนใจแบบไม่เป็นเชิงเส้น (The Nonlinear Activation Function)

การนำเอาท์พุทของเลเยอร์หนึ่ง มาเชื่อมกับอินพุทของเลเยอร์ชั้นถัดไป โดยผ่านฟังก์ชั่นกระตุ้นความสนใจแบบไม่เป็นเชิงเส้น จะทำให้โครงข่ามีความสามารถในการคำนวณเพิ่มขึ้น (หากไม่ผ่านฟังก์ชั่นดังกล่าวความสามารถการคำนวณจะไม่เพิ่มขึ้น และจะมีความสามารถไม่แตกต่างไปจาก Single layer networks) และสามารถกำหนดขอบเขตของเอาท์พุทให้อยู่ในช่วงที่ต้องการได้

2.11.7 การฝึกสอนให้กับโครงข่ายประสาทเทียม (Training of Artificial Neural Networks)

ค่าน้ำหนัก มีความสัมพันธ์กับอะไร เปลี่ยนแปลงอย่างไรนั้น ก็เช่นเดียวกับเด็กที่คลอดออกมาก็มีสมองแล้ว แต่สมองยังไม่เจริญเติบโตเพียงพอ และยังไม่ได้รับการฝึกสอนและเรียนรู้ เด็กจึงไม่สามารถทำกิจกรรมใดๆด้วยตนเอง เว้นแต่กิจกรรมที่ธรรมชาติสร้างมาพร้อมกับการกำเนิดที่เรียกว่า "สัญชาติญาณ" ซึ่งธรรมชาติใส่คุณลักษณะบางอย่างให้เซลล์สมองบางส่วนตั้งแต่ทารกเจริญเติบโตอยู่ในครรภ์มารดา เช่น ระบบควบคุมการหายใจ, ความรู้สึก, การเรียกร้องเมื่อหิว, การตอบสนองต่อสิ่งเร้า ฯลฯ เด็กจะพัฒนาการเรียนรู้ไปตามขั้นตอน หลังจากนั้นสมองของเขาจะได้รับการฝึกสอน และเจริญเติบโตไปพร้อมๆกัน เซลล์สมองจะได้รับการปรับคุณลักษณะสอดคล้องกับการฝึกสอน และจะเจริญเป็นโครงข่ายสอดคล้องกัน

โครงข่ายประสาทเทียมที่สร้างขึ้นมีลักษณะเช่นเดียวกัน คือ เมื่อสร้างเสร็จ แต่ละเซลล์ประสาทที่สร้างขึ้นมานั้นจะยังไม่มีคุณลักษณะใดเลย เนื่องจากยังไม่มีการกำหนดค่าซินแนปติกส์เวจที่เหมาะสมกับงานที่ต้องการให้กับมัน จึงต้องมีการฝึกสอนเพื่อให้เน็ตเวอร์คที่สร้างขึ้นมีลักษณะตามที่ต้องการ การฝึกสอนของโครงข่ายประสาทเทียม จะกระทำโดยการปรับเปลี่ยนค่าน้ำหนักซินแนปติกส์ เพื่อให้โครงข่ายจดจำ แพตเทอร์น ความสัมพันธ์ระหว่างอินพุทกับเอาท์พุทได้ โดยในขั้นแรกอาจกำหนดเป็นค่าสุ่มใดๆ (Random weight) ก่อนแล้วถึงปรับเปลี่ยนน้ำหนักไปตามอัลกอริทึมสมมติฐานหลายๆรอบ จนกว่าจะได้เอาท์พุทที่ต้องการ ในเงื่อนไขความผิดพลาดที่ยอมรับได้

2.11.8 วัตถุประสงค์ของการเทรนนิ่ง (Objective of Training)

เนื่องจากค่าน้ำหนักที่ให้เป็นค่าสุ่มใดๆ โครงข่ายจึงไม่แสดงคุณลักษณะใดออกมา การฝึกสอน (Training) ให้โครงข่ายก็คือ การปรับค่าน้ำหนักทุกๆจุด ให้สอดคล้องกัลป์อินพุทหลายๆแบบ เพื่อให้ได้เอาท์พุทตามต้องการนั่นเอง การฝึกสอนโครงข่ายจะต้องบรรลุถึงกระบวนการเข้าใจพื้นฐานเสียก่อน คือ การเรียนรู้ในโครงข่ายประสาทเทียมนั้นมีขีดจำกัด ปัญหาต่างๆ ผู้ใช้คงต้องแก้ให้มันก่อน แล้วนำผลนั้นไปอ้างอิงสำหรับการปรบปรุงค่าน้ำหนัก หลังจากปรับน้ำหนักจนได้ค่าผิดพลาดที่เอาท์พุทเทียบกับเป้าหมายน้องลงเป็นที่พอใจแล้ว โครงข่ายประสาทเทียมนั้นก็พร้อมที่จะวิเคราะห์อินพุท และให้เอาท์พุทตามลักษณะตัวอย่างที่มันเคยเรียนรู้มา การเรียนรู้จะมีการปรับน้ำหนักหลายๆรอบ จนค่าน้ำหนักสอดคล้องกับตัวอย่างหลายๆตัวอย่าง และให้เอาท์พุทตามต้องการ

พบว่าโครงข่ายได้ตัวอย่างสำหรับการเทรนนิ่งมากๆ โครงข่ายก็จะมีความแม่นยำสูงขึ้น แต่ก็จะใช้เวลาในการเทรนนิ่งเพิ่มขึ้นเช่นกัน หากพิจารณาต่อไปจะพบว่า โครงข่ายประสาทเทียมที่สร้างขึ้นจะมีพฤติกรรมคล้ายกับระบบการเรียนรู้ของมนุษย์มาก เป็นเพราะมีต้นแบบมาจากระบบประสาทชีวภาพนั่นเอง

2.11.9 การเทรนนิ่งแบบควบคุม (Supervised Training)

เทรนนิ่งอัลกอริทึมถูกจัดเป็น 2 ประเภท คือ แบบควบคุม (Supervised training) และแบบอิสระ (Unsupervised training) โดยการเทรนนิ่งแบบควบคุม จะต้องการคู่ของเทรนนิ่งระหว่างอินพุทกับเป้าหมายที่ต้องการที่เรียกว่า เทรนนิ่งแพร์ (Training pairs) โครงข่ายจะถูกเทรนไปตามจำนวนของคู่ที่เทรนนิ่ง (จำนวนคู่ของ Input กับ Output ที่ต้องการให้โครงข่ายรู้จัก) เอาท์พุทที่คำนวณได้จากโครงข่ายจะถูกเปรียบเทียบกับความสอดคล้องกับเป้าหมาย ค่าผิดพลาดที่เกิดขึ้นจะถูกป้อนกลับไปยังโครงข่าย และเปลี่ยนแปลงค่าน้ำหนักให้สอดคล้องกับอัลกอริทึม ที่ทำให้แนวโน้มของค่าผิดพลาดที่เกิดขึ้นระหว่างเอาท์พุทกับเป้าหมายโดยเฉลี่ยมีค่าลดต่ำลง ตัวอย่างการเทรนนิ่งแบบนี้ ได้แก่ การเทรนนิ่งแบบแพร่กลับ (Back propagation)

2.11.10 การเทรนนิ่งแบบอิสระ (Unsupervised Training)

ถึงแม้ว่า อัลกอริทึมแบบควบคุม (Supervised training) สามารถจะประยุกต์ใช้เพื่อปรับคุณลักษณะของโครงข่ายได้สำเร็จ แต่ก็ยังมีข้อวิจารณ์อยู่ คือ มันเป็นไปอย่างแบบชีวภาพไม่ได้ และยากที่จะเชื่อได้ว่า กลไกการเทรนนิ่งของสมองจะต้องการการเปรียบเทียบระหว่างคนที่ต้องการกับเอาท์พุทจริง โดยการบวนการป้อนกลับไปแก้ไขคุณลักษณะของโครงข่าย และถ้าสมมติว่า ถ้าสมองมีกลไกเช่นนี้ ต้องมีเอาท์พุทที่ต้องการเพื่อนำมาเป็นเป้าหมายตลอดเวลา และจะเอามาจากที่ใด สรุปคือต้องมีผู้คิดเป้าหมายให้กับโครงข่ายก่อน โครงข่ายไม่สามารถคิดและปรับคุณลักษณะได้ก่อนด้วยตนเอง ในทางตรงกันข้ามหากพิจารณาทารกแรกเกิดสมองของเขาสามารถจัดระบบเองได้อย่างไร การเทรนนิ่งแบบระบบของสมอง จนกระทั่งมีการพัฒนาการเทรนนิ่งแบบอิสระนี้ขึ้นราวปี 1984 โดย โคโฮเนน (Kohonen) และบุคคลอื่นๆ โดยได้เสนอแนวคิดที่เป็นการเทรนนิ่งแบบไม่ต้องการเป้าหมาย ไม่มีการตัดสินใจด้วยเหตุผลในอุดมคติมาก่อน ชุดของการเทรนนิ่ง จะมีเพียงอินพุทเวกเตอร์เท่านั้น เทรนนิ่งอัลกอริทึมจะเปลี่ยนแปลงค่าน้ำหนักของโครงข่าย เพื่อสร้างเอาท์พุทที่มีความมั่นคง ยกตัวอย่างเช่น หากให้โครงข่ายรู้จำภาพหน้าคนหนึ่ง หากภาพหน้าคนคนนั้นเปลี่ยนแปลงไปเล็กน้อย (Image อาจมี Noise ร่วมอยู่บ้าง) โครงข่ายนั้นก็ยังสามารถบอกได้ว่า คนคนนั้นเป็นคนเดิมเป็นต้น การเทรนนิ่งจะไม่มีการตัดสินใจมาก่อน ไม่มีการกำหนดแบบเอาท์พุทมาก่อน (อาจกล่าวได้ว่าแบบเอาท์พุทจะถูกกำหนดโดยอินพุทเวกเตอร์นั่นเอง) ดังนั้น เอาท์พุทของโครงข่ายก็เช่นกัน ส่วนใหญ่จะถูกแปรรูปซึ่งจะเข้าใจได้ภายหลังกระบวนการเทรนนิ่ง ดังนั้นจึงไม่สามารถ

แก้ปัญหาที่เคร่งครัดสำคัญได้ แต่มักนิยมใช้โครงข่ายแบบนี้กับงานง่ายๆ ประเภทการเปรียบเทียบเอกลักษณ์, รูปแบบที่สัมพันธ์กันระหว่างอินพุท-เอาท์พุท ที่ถูกกำหนดโดยโครงข่าย

2.11.11 วิธีการแก้ปัญหาการฝึกสอน (Training Algorithm)

ส่วนใหญ่แล้วทุกวันนี้ การแก้ปัญหาฝึกสอนของโครงข่ายค่อยๆพัฒนาก้าวหน้าขึ้นจากแนวความคิดของ ดี โอ เฮบบ์ (ปี 1961) เขาได้เสนอโมเดลของ การเทรนนิ่งแบบอิสระ (Unsupervised training) ในแบบซินแนปติกส์สเตรงท์หรือน้ำหนัก ซึ่งจะเพิ่มขึ้น ถ้าทั้งแหล่งกำเนิด (Input Source) และจุดหมายปลายทาง (Destination) ของนิวรอลได้รับการสนใจ กรณีนี้ถ้ามีการใช้งานทางเส้นนี้บ่อยๆก็จะทำให้ซินแนปติกส์สเตรงจ์ (หรือน้ำหนัก) แข็งแรงขึ้น (เซลล์สมองที่ใช้งานมากบ่อยๆ ก็จะทำให้ ซินแนปส์ใหญ่ขึ้น การส่งผ่านข้อมูลพัลส์ไฟฟ้าทำได้ดีขึ้น ทำให้มีประสิทธิภาพดีขึ้น เช่น สามารถคิด หรือจดจำได้เร็ว และดีขึ้น) โครงข่ายประสาทเทียมนี้ ใช้การเรียนรู้แบบเฮบเบน (Hebbian learning) จะเพิ่มค่าน้ำหนักของโครงข่ายอย่างสอดคล้องกับผลคูรของระดับความสนใจของแหล่งกำเนิด และจุดหมายของนิวรอลตามสมการดังนี้

$$W_{ij}(n+1) = W_{ij}(n) + \alpha OUT_i OUT_j \qquad (2.17)$$

โดย $W_{ij}(n)$ คือค่าน้ำหนัก จากนิวรอล I ไปยังนิวรอล j ก่อนปรับปรุงค่า

$W_{ij}(n+1)$ คือค่าน้ำหนัก จากนิวรอล I ไปยังนิวรอล j หลังปรับปรุงค่า

α คือค่าคงที่ของการเรียนรู้ (Learning rate coefficient)

OUT_i คือเอาท์พุทของนิวรอล I และเป็นอินพุทของนิวรอล j

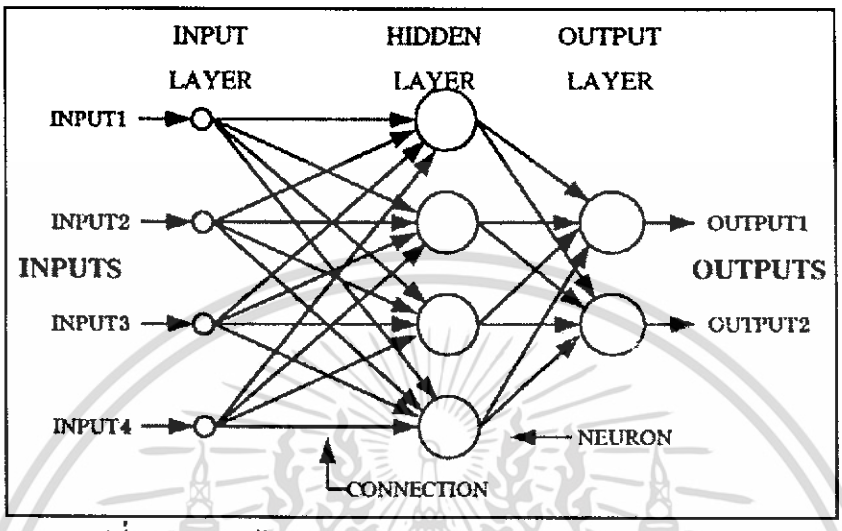
OUT_j คือเอาท์พุทของนิวรอล j

โครงข่ายที่มีลักษณะการเทรนนิ่งแบบเฮบเบนนั้น เป็นผลมาจากการพัฒนามาแล้วกว่า 20-30 ปี โดยเฉพาะงานของโรเซนเบลทท์ (Rosenblatt:1962), วิโดรว์ (Widrow:1959), วิโดรว์และฮอฟฟ์ (Widrow&Holff:1960) และอีกหลายๆคนที่พยายามพัฒนาระบบการเทรนนิ่งแบบควบคุม ที่สร้างโครงข่ายที่สามารถเรียนรู้แบบของอินพุทได้อย่างกว้างขวาง และมีอัตราการเรียนรู้สูง ที่บรรลุผลได้จากหลักการพื้นฐานของการเรียนรู้ หรือเทรนนิ่งให้กับโครงข่ายเช่น Perceptrons, Hopfieldnets, Backpropagation Networks และ Counter propagation.

2.11.12 การประยุกต์ใช้นิวรัลเน็ตเวอร์คกับปริญญานิพนธ์

จากการศึกษาพฤติกรรม การสื่อภาษาของมนุษย์พบว่า การเรียนรู้ภาษาต้องเรียนรู้จากผู้อื่น ภาษาไม่ใช่สิ่งที่เป็นสัญชาติญาณ ดังนั้นจึงต้องเรียนรู้จากภาษาที่สังคมนั้นกำหนดขึ้น ซึ่งเป็นแพต

เทอร์นของภาษาที่บรรพบุรุษได้สร้างขึ้นสั่งสมกันมาก่อน จากเหตุผลดังกล่าว พบว่าสอดคล้องกับอัลกอริทึม การเทรนนิ่งแบบควบคุม (Supervised training) ผู้ทำจึงเลือกใช้วิธี Backpropagation มัลติเลเยอร์เป็นแบบ 2 เลเยอร์ แสดงดังภาพที่ 2.19

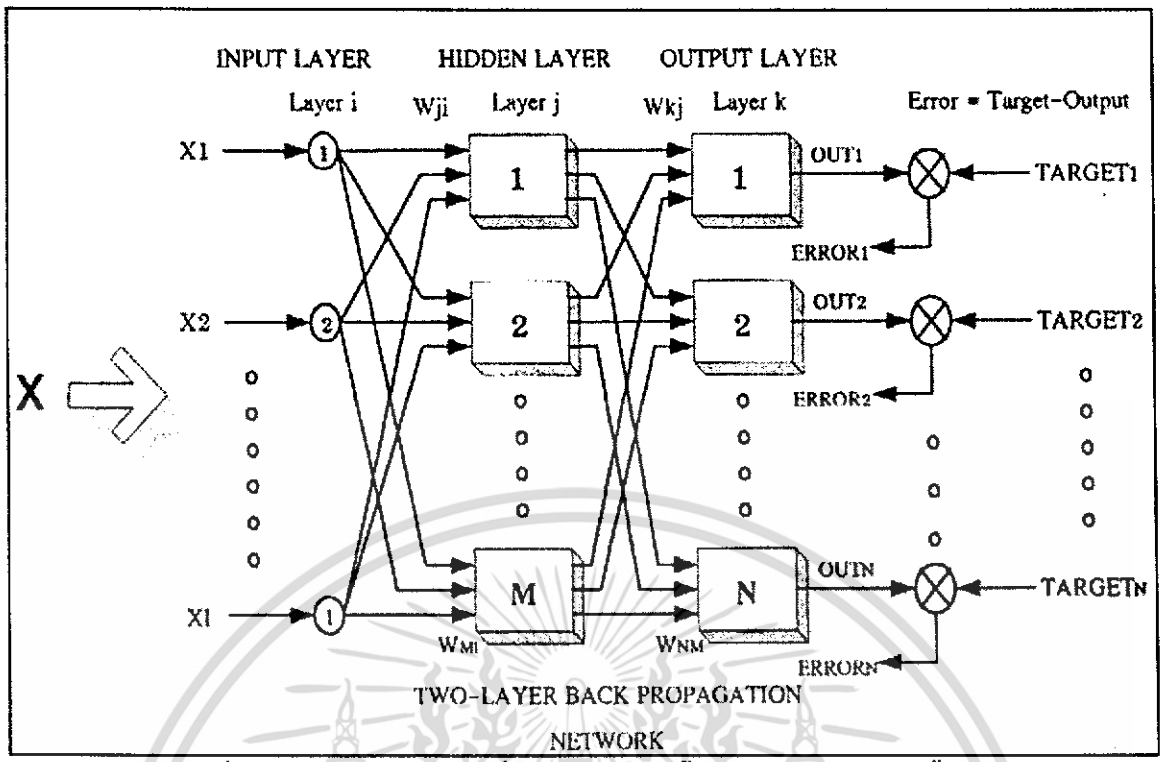


รูปที่ 2.19 แสดงไดอะแกรมของ Two Layer Feed - Forward

พิจารณาภาพที่ 2.19 ประกอบ จะเห็นว่า ประกอบด้วย input array, hidden layer และ output layer ในส่วนของ Input array จะไม่ถือเป็น layer เพราะไม่มีการคำนวณใดๆ แต่มันจะทำหน้าที่เชื่อมโยงข้อมูลให้กับ Hidden layer ทุกๆนิวรอล output ของ hidden layer ก็จะถูกเชื่อมโยงไปเป็น input ให้กับ output layer ทุกอินพุทของนิวรอลทุกตัวจะมีค่า Weight ประจำตัว ค่า Weight นี้ทำให้ระดับของอินพุทผ่านเข้าไปภายในนิวรอลแต่ละตัวไม่เท่ากัน ซึ่งเป็นตัวกำหนดคุณลักษณะสมบัติของนิวรอลแต่ละตัวในเน็ตเวอร์ค วิธีการกำหนดคุณสมบัติดังกล่าว ทำได้โดยการเทรนนิ่งโดยในขั้นแรก จะสมมติว่า ค่าแบบสุ่มน้อยๆให้ (ช่วงที่ใช้คือตั้งแต่ -0.5 ถึง +0.5) แล้วจึงค่อยๆปรับเปลี่ยนค่าน้ำหนักในแต่ละ layer ให้สอดคล้องกับคุณสมบัติที่ต้องการ

2.11.13 กระบวนการเทรนนิ่ง

กระบวนการเทรนนิ่ง คือ กระบวนการปรับคุณลักษณะของนิวรัลเน็ตเวอร์ค ให้มีคุณสมบัติตามที่ต้องการ การเทรนนิ่งของอัลกอริทึม Backpropagation เป็นแบบ Supervised Training แบ่งเป็น 2 ส่วนคือ Forward pass และ Reversed pass การเทรนนิ่งจะกระทำทั้ง Forward pass และ Reversed pass สลับกันไป จนกว่าค่าผิดพลาดระหว่าง Output กับ Target จะมีค่าต่ำกว่าจุดที่ต้องการ



รูปที่ 2.20 แสดงไดอะแกรมเชื่อมโยงระหว่างชั้นของนิวรอลแบบ 2 ชั้น ของ Backpropagation Network

FORWARD PASS

จากภาพ 2.20 อินพุท X ตั้งแต่ X_1 ถึง X_I จะถูกคูณด้วย Weight matrix ผลรวมของอินพุทเมื่อคูณกับค่า Weight ของแต่ละนิวรอลเรียกว่า เน็ต (NET) เขียนเป็นสมการได้คือ

$$NET_j = X_1W_{j1} + X_2W_{j2} + ... + X_IW_{jI} \qquad (2.18)$$

หรือ

$$NET_j = \sum_{i=1}^{I} X_iW_{ij} \qquad (2.19)$$

โดย X คือ ค่าที่รับเข้ามาของ Input node

W คือ ค่าน้ำหนักประจำตัวของแต่ละอินพุทโหนด

I คือ หมายเลขโหนดโดยเริ่มที่โหนด 1 ถึง โหนด I

J คือ หมายเลขนิวรอลใน Hidden layer ตั้งแต่นิวรอลที่ 1 ถึง M

K คือ หมายเลขนิวรอลใน Output layer ตั้งแต่นิวรอลที่ 1 ถึง N

และทำนองเดียวกัน

$$NET_k = \sum_{j=1}^{M} OP_j W_{kj} \tag{2.20}$$

เมื่อได้ค่า NET ของแต่ละนิวรอลแล้ว จะปรับให้ได้ Output อยู่ในช่วงที่ต้องการเป็นค่าที่จะนำไปยับยั้ง หรือกระตุ้นนิวรอลตัวถัดไปโดย Squashing Function ที่ใช้เป็น Logistic Function หรือ Sigmoid จะได้ตามสมการ 2.21 และ 2.22

$$OP_j = F_j(NET_j) = \frac{1}{1+e^{-(net/\theta_j)}} \tag{2.21}$$

$$OP_k = F_k(NET_k) = \frac{1}{1+e^{-(net/\theta_k)}} \tag{2.22}$$

กำหนด $\theta_j = 0.05$ และ $\theta_k = 0.05$

θ_j และ θ_k คือ ค่าที่กำหนดความชันของ Sigmoid curve

OP_j คือ ค่า Output ที่ Neutron j ของ Hidden Layer (เป็น Input ของ Output Layer)

OP_k คือ ค่า Output ที่ Neutron k ของ Hidden Layer (เป็น Input ของ Output Layer)

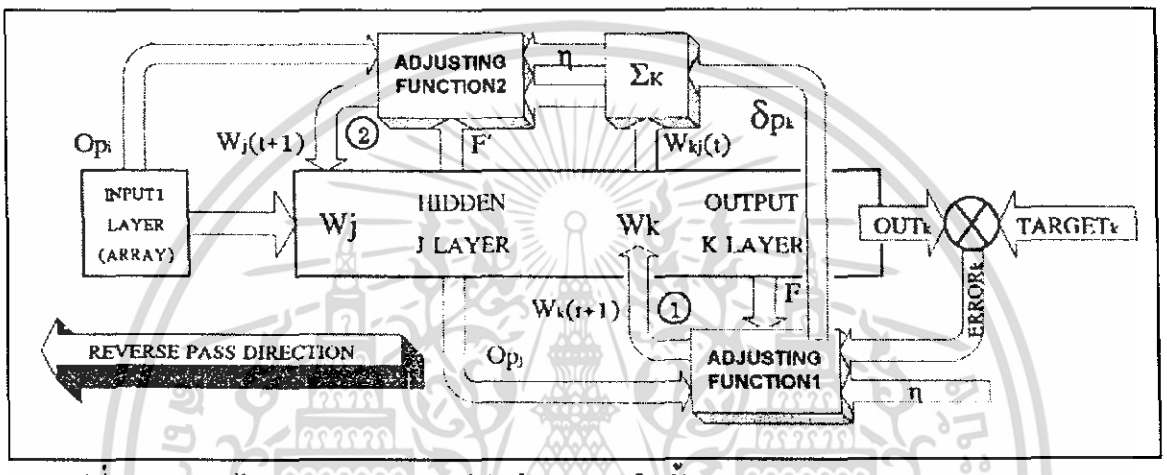
การคำนวณหาค่า Output จะเริ่มที่ Hidden layer ที่ชั้น j ก่อน เมื่อได้ค่า output ของ Hidden layer ครบทุกตัวแล้ว จึงคำนวณหาค่า Output ที่ Output layer โดย Input ของ Output layer ก็คือ Output ของ Hidden layer นั่นเอง โดยใช้การคำนวณทำนองเดียวกันกับ Hidden layer Output ที่ได้จะนำไปเปรียบเทียบกับ Target ที่ต้องการเกิดเป็น Error ขึ้น ค่า Error นี้ จะถูกนำไปใช้ร่วมกับค่าดิฟเฟอร์เรนเชียล ของ F(F′) ใน Output layer เพื่อปรับค่า Weight ให้กับ Output layer ดังจะกล่าวต่อไปในส่วนของ Reverse pass

REVERSE PASS

เป็นส่วนที่ป้อนกลับค่า Error(Target-Output) จากเอาท์พุทเลเยอร์ย้อนกลับมาปรับปรุงค่าน้ำหนักที่เลเยอร์ก่อนหน้า สำหรับนิวรัลเน็ตเวอร์คที่มีมากกว่า 1 เลเยอร์ การปรับปรุงค่าน้ำหักชั้นถัดถอยมา จะใช้ค่าน้ำหนักเดิมของเลเยอร์ที่สูงกว่า ร่วมกับตัวร่วมทางด้าน Output ของเลเยอร์มาคำนวณ (เนื่องจากไม่มีค่า Error มาใช้เหมือน Output layer) ดังจะกล่าวในรายละเอียดดังต่อไปนี้

จากภาพที่ 2.21 การปรับเปลี่ยนค่าน้ำหนักใหม่ จะเป็นไปตามทิศทางของลูกศร และได้ข้อมูลจากค่า Error ที่ Output layer เป็นต้นกำเนิดเพื่อกำหนดทิศทางปรับค่าน้ำหนักให้นิวรัลเน็ตเวอร์ค มีคุณลักษณะสมบัติตามต้องการ ลำดับขั้นตอนของ Reverse pass พอสรุปได้ดังนี้คือ

1. นำ Output layer ไปลบออกจาก Target ได้เป็น Error
2. นำค่า Error มาคูณกับ F′ ที่ได้จาก Output layer เรียกว่า δp_k
3. นำค่า δp_k คูณกับ Op_j แล้วคูณกับ η (อัตราการเรียนรู้ 0.01-1)
4. ค่า Weight ค่าใหม่ที่ได้ คือ ค่า Weight เดิมบวกกับผลลัพธ์ ที่ได้จากข้อ 3



รูปที่ 2.21 แสดงไดอะแกรมของการปรับค่า Weight ในชั้น Output layer และ Hidden layer

พอเขียนเป็นสมการใหม่ได้คือ

$$W_{kj}(t+1) = W_{kj}(t) + \eta \delta p_k Op_j \qquad (2.23)$$

เพราะว่า $$\eta \delta p_k Op_j = \Delta W_{kj} \qquad (2.24)$$

ฉะนั้น $$W_{kj}(t+1) = W_{kj}(t) + \Delta W_{kj} \qquad (2.25)$$

เมื่อ $W_{kj}(t)$ คือ ค่าน้ำหนักเดิมก่อนปรับ

$W_{kj}(t+1)$ คือ ค่าน้ำหนักหลังปรับ

η คือ อัตราการเรียนรู้

Op_j คือ Output ของ Layer j

δp_k คือ ค่าที่ได้จาก Error คูณกับดิฟเฟอเรนเชียลของฟังก์ชั่น F

การปรับค่า Layer ถัดมา (Hidden layer) มีลักษณะคล้ายคลึงกับ Output layer คือ

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_{p_j} Op_i \quad (2.26)$$

เพราะว่า $$\eta \delta_{p_j} Op_i = \Delta W_{ji} \quad (2.27)$$

ฉะนั้น $$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji} \quad (2.28)$$

สมการที่ 2.27 แตกต่างกันกับสมการ 2.24 ตรงการได้มาซึ่ง δ_{p_j} เพราะไม่มีค่า Output กับ Target ที่ Hidden layer ค่าดังกล่าวสามารถทดแทนได้ด้วย $\sum_k(\delta_{p_k} W_{kj})$

โดย $$\delta_{p_j} = Op_j(1 - Op_j)(\sum_k(\delta_{p_k} W_{kj})) \quad (2.29)$$

เมื่อ $W_{ji}(t)$ คือ ค่าน้ำหนักเดิมก่อนปรับ

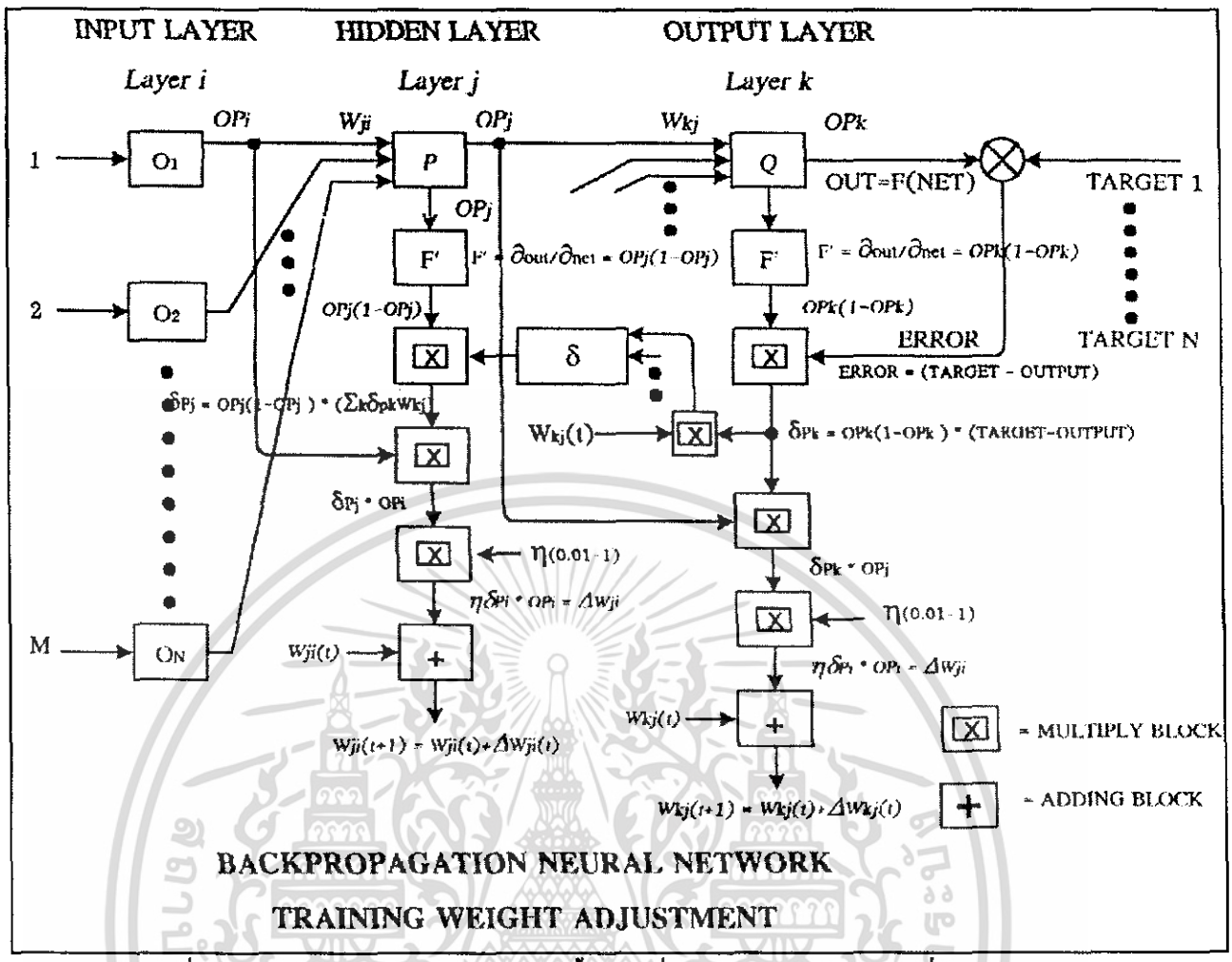
$W_{ji}(t+1)$ คือ ค่าน้ำหนักหลังปรับ

η คือ อัตราการเรียนรู้

Op_i คือ ข้อมูลที่ได้จาก Input Layer

δ_{p_j} คือ ค่าควบคุมให้น้ำหนักเปลี่ยนแปลงเข้าสู่จุดดีที่สุด

การปรับค่าน้ำหนักจะปรับทีละเลเยอร์จาก Output layer กลับไปยัง Input layer เมื่อเสร็จสิ้นก็จะกลับสู่ส่วนของ Forward pass อีกครั้ง เพื่อหา Output มาเปรียบเทียบกับ Target และดำเนินการปรับค่าน้ำหนักในส่วนของ Reverse pass เช่นนี้สลับกันไปจนกว่าค่า Error ที่ได้จะลดลงต่ำกว่าค่าที่ต้องการจึงหยุดกระบวนการ ค่าน้ำหนักที่ได้จะเป็นค่าที่เหมาะสมสำหรับทุกคู่ของ Input และ Target ซึ่งพร้อมที่จะนำไปใช้ในกระบวนการตรวจสอบ การปรับค่าน้ำหนักเทรนนิ่งของ Backpropagation Network พออธิบายได้ด้วยไดอะแกรม ดังนี้



รูปที่ 2.22 ไดอะแกรมแสดงวิธีการหาค่าน้ำหนักที่เหมาะสมของการเทรนนิ่ง โดยใช้อัลกอริทึมของแบคพรอบพาเกชั่น

2.11.14 เงื่อนไขการหยุดฝึกสอน

การกำหนดเงื่อนไขการหยุดฝึกสอน ทำได้สองกรณี คือ

1. เมื่อผลรวมของค่าผิดพลาดเฉลี่ยระหว่างเอาท์พุตกับเป้าหมายทั้งหมด (SSQERR) ลดน้องลง กว่าค่าที่กำหนด โดยกำหนดสมการหาค่าผิดพลาดเฉลี่ยของเอาท์พุตทั้งหมดดังนี้

$$SSQERR = \frac{1}{2}\sum (T\arg et - Output)^2 \qquad (2.30)$$

SSQERR คือ ค่า Sum Square Error ที่ได้จาก Output neurons เทียบกับ Target ใช้สำหรับบ่งชี้ผลของการฝึกสอนว่าเพียงพอกับการใช้งาน หรือเงื่อนไขที่กำหนดหรือไม่

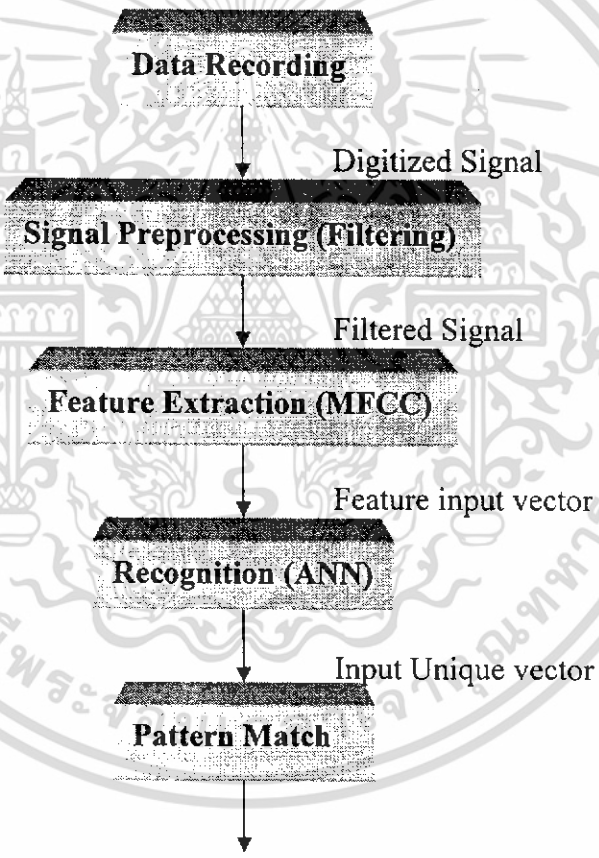
2. จำนวนรอบของการฝึกสอนดำเนินไปถึงค่าที่ตั้งไว้ ซึ่งจำนวนรอบที่เหมาะสมได้จากการทดลอง (จากที่ศึกษา จำนวนรอบที่เหมาะสมอยู่ในช่วง 2,500 – 5,000 รอบ)

บทที่ 3
การออกแบบและพัฒนา

3.1 บทนำ

การทำงานของระบบในปริญญานิพนธ์ประกอบด้วย 3 ส่วนหลัก คือ

- การประมวลผลเบื้องต้น (Preprocessing)
- การสกัดค่าลักษณะสำคัญ (Feature extraction)
- การรู้จำ (Recognition)



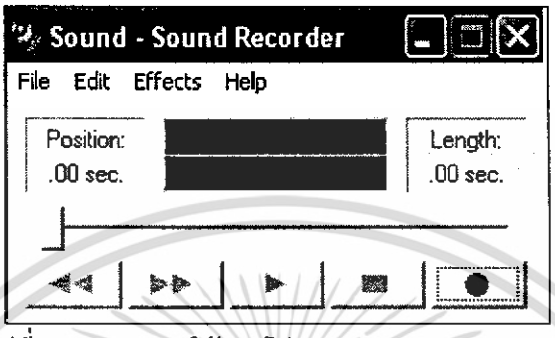
รูปที่ 3.1 แสดงองค์ประกอบหลักของระบบรู้จำเสียง

3.2 โปรแกรมที่ใช้ในระบบ

โปรแกรมทั้งที่ใช้ในระบบประกอบไปด้วย

3.2.1 ส่วนการอัดเสียง

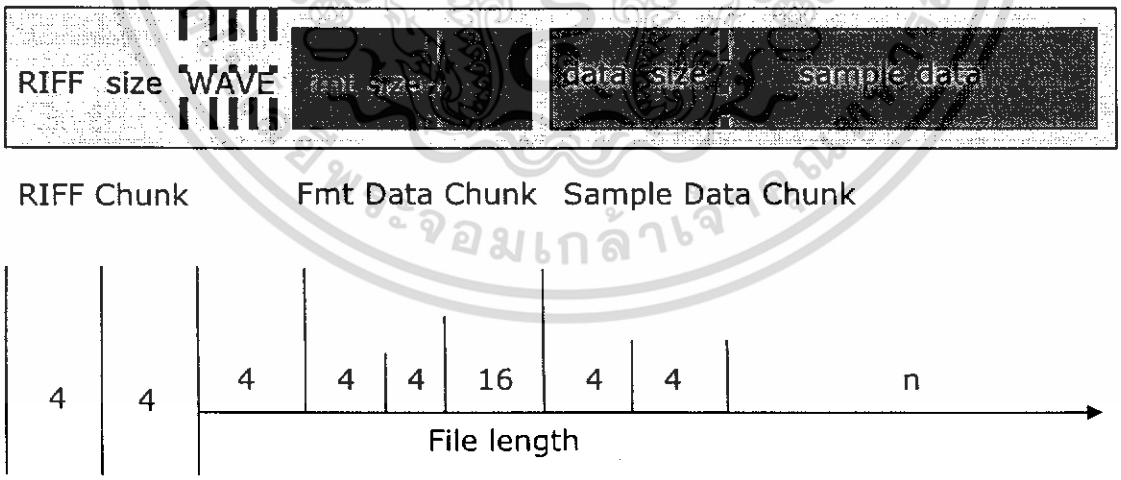
ทำโดยใช้โปรแกรม Sound Recorder ที่มีมาพร้อมกับ Windows XP แล้ว



รูปที่ 3.2 แสดงการใช้งานโปรแกรม Sound Recorder

ในการบันทึกเสียงพูดเป็นข้อมูลดิจิตอลโดยใช้การ์ดเสียง จะทำให้ได้เวฟไฟล์ (.wav) ที่มีรูปแบบของไฟล์ RIFF (Resource International File Format) ซึ่งเป็นรูปแบบมาตรฐานของ MS windows ที่ใช้กันอย่างกว้างขวาง

ไฟล์ RIFF นั้นมีโครงสร้างเชิงซ้อนคือ มีลักษณะเป็นกลุ่ม (Chunk) ซึ่งภายในกลุ่มนี้ยังประกอบด้วยกลุ่มย่อยซึ่งมีโครงสร้างคล้ายกัน ดังแสดงในรูปที่ 3.3



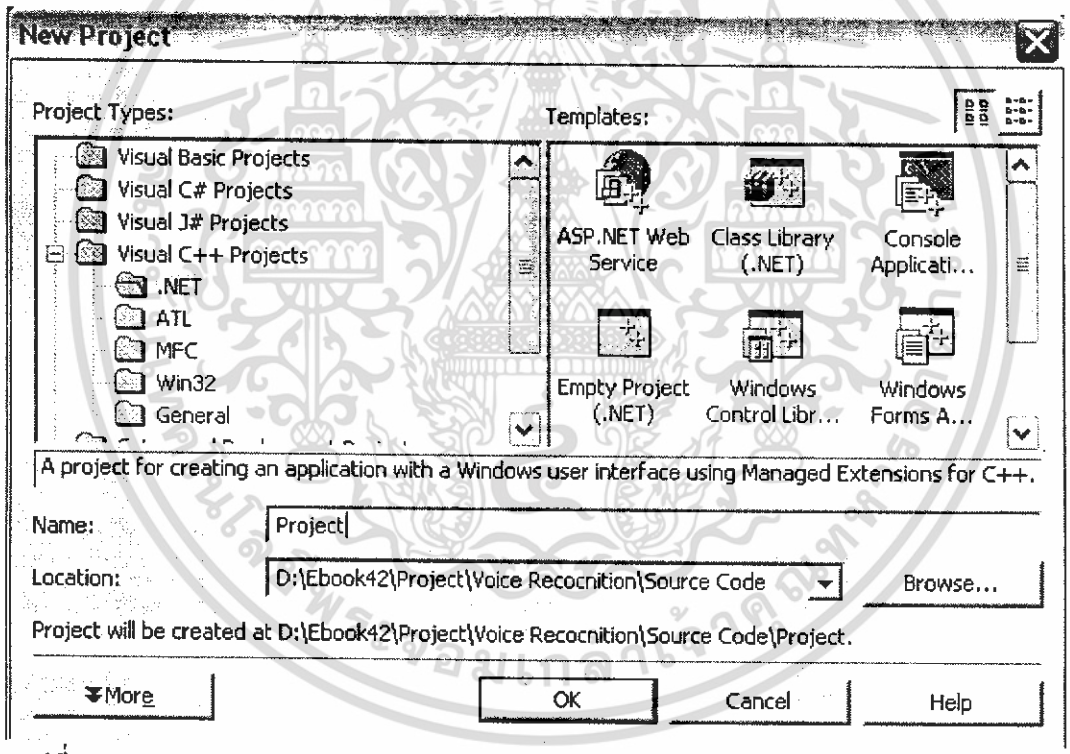
รูปที่ 3.3 แสดงโครงสร้างเวฟไฟล์ที่มีรูปแบบของไฟล์ RIFF

จากรูปที่ 3.3 จะเห็นว่า กลุ่ม RIFF (RIFF Chunk) จะเป็นกลุ่มใหญ่ที่สุด ซึ่งประกอบด้วยกลุ่มรูปแบบของข้อมูล (Format Data Chunk) และกลุ่มข้อมูลจากการแซมปลิ้ง (Sample Data Chunk)

ในการทดลอง กำหนดให้เก็บเสียงในรูปแบบของไฟล์ WAV อัตราการชักตัวอย่าง (Sampling rate)ที่ 11.025 กิโลเฮิร์ต ตัวอย่างละ 16 บิต และแบบช่องสัญญาณเดียว(Mono)

3.2.2 ส่วนประมวลผลและพัฒนา

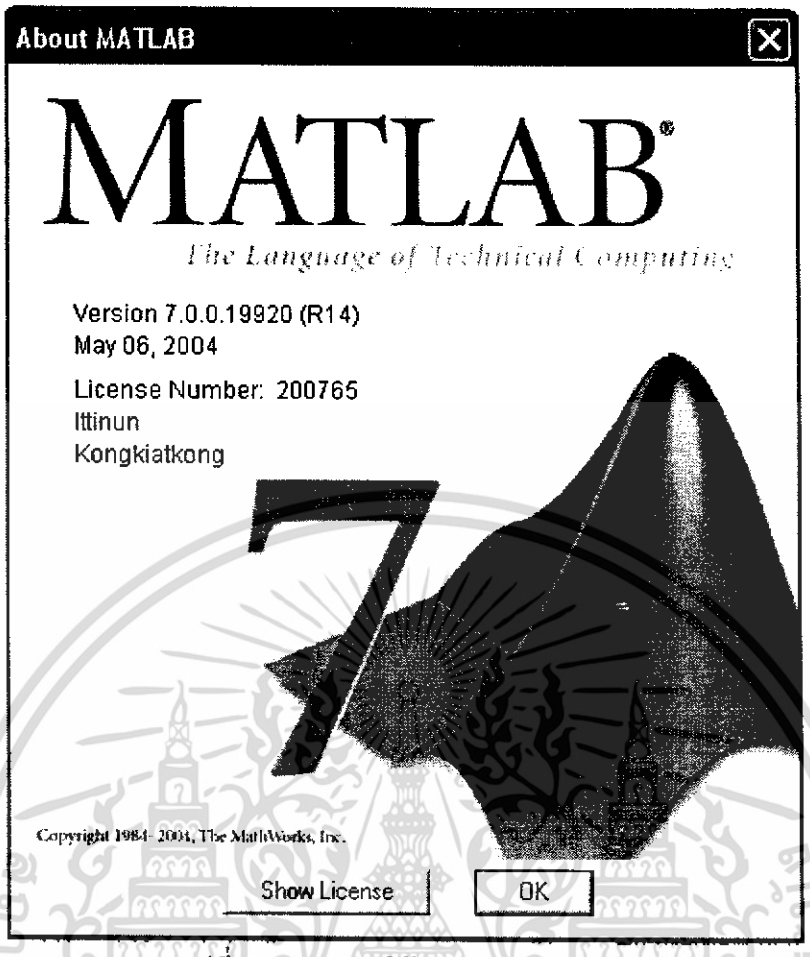
ทำการประมวลผลและพัฒนา พัฒนาด้วยโปรแกรม Microsoft Visual C++ .NET พัฒนาด้วยรูปแบบ C# เพื่อให้สะดวกต่อการนำไปรวมกับส่วนโปรแกรมที่เกี่ยวข้องโดยออกแบบให้ส่วนนี้เป็นคลาสแล้วเรียกใช้ผ่านเมธอดของออบเจ็คแทนการเรียกใช้แบบฟังก์ชัน ทำให้การแก้ไขและพัฒนาโมดูลทำได้ง่ายและสะดวก



รูปที่ 3.4 แสดงการใช้งาน Microsoft Visual C++ .NET แบบ Windows Forms Application

3.2.3 ส่วนแสดงผล

ทำการทดสอบทฤษฎีและการทำงานโดยใช้ MATLAB 7.0 เนื่องจากมีฟังก์ชั่นสำหรับการแปลงสัญญาณเสียง มีกราฟแสดงรูปแบบของสัญญาณเสียง และง่ายต่อการเรียนรู้และใช้งาน



รูปที่ 3.5 แสดงการใช้งาน MATLAB 7.0

3.3 การประมวลผลเบื้องต้น (Preprocessing)

สัญญาณเสียงที่ผ่านการแปลงสัญญาณเป็นดิจิตอลแล้ว จะนำมาผ่านขั้นตอนการประมวลผลเบื้องต้น ซึ่งประกอบด้วยขั้นตอนต่างๆ ดังนี้

3.3.1 การกรองทางความถี่ (Filtering)

เป็นขั้นตอนในการกรองสัญญาณในช่วงความถี่ที่ไม่ต้องการออกโดยอาศัยตัวกรองแบบดิจิตอล โดยใช้ตัวกรองแบบดิจิตอลชนิดผ่านความถี่สูง กำหนดจุดผ่านความถี่ (Cutoff frequency) ที่ 200 เฮิร์ต เพื่อป้องกันสัญญาณรบกวนความถี่ต่ำที่เกิดจากแหล่งกำเนิดไฟฟ้า

3.3.2 การตัดหัว-ท้ายเสียง (Endpoint detection)

เป็นขั้นตอนในการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของเสียงโดยการแยกส่วนที่เป็นคำพูดออกจากส่วนวนที่ไม่ใช่คำพูด วิธีในการตัดหัว-ท้ายเสียงหลายวิธี เช่น ใช้ค่าระดับพลังงาน (Energy level) ใช้อัตราการตัดศูนย์ (Zero-crossing rate) เป็นต้น

3.3.3 การนอร์มอลไลซ์ทางเวลา (Time normalization)

เป็นขั้นตอนการเพิ่มหรือลดขนาดความยาวของสัญญาณในเชิงเวลา เพื่อปรับแต่งขนาดความยาวของสัญญาณให้เหมาะสมตามต้องการทั้งนี้จะขึ้นอยู่กับกระบวนการในการรู้จำว่าจำเป็น

ต้องทำการนอร์มอไลซ์สัญญาณให้เท่ากันหรือไม่ วิธีในการนอร์มอลไลซ์ทางเวลามีหลายวิธี เช่น การเปลี่ยนอัตราการชักตัวอย่าง (Sampling rate changing) การประมาณค่าในช่วงเชิงเส้น (Linear interpolation) และการเหลื่อมและรวมส่วนย่อยแบบซิงโครไนซ์ (Synchronized overlap-and-add) เป็นต้น

3.4 การสกัดค่าลักษณะสำคัญ (Feature)

การสกัดค่าลักษณะสำคัญ คือการวิเคราะห์หาค่าที่จะใช้แทนสัญญาณเสียง เพื่อนำไปใช้ในขั้นตอนการรู้จำ แบ่งได้เป็น 3 กลุ่มหลัก กลุ่มแรกเป็นค่าลักษณะสำคัญระดับสูง (High level feature) ได้แก่ สำเนียงการพูด รูปแบบในการพูด และความเร็วในการพูด เป็นต้น ในกลุ่มที่สอง จะใช้ค่าลักษณะสำคัญทางฉันทลักษณ์ (Prosodic feature) เช่น ค่าความถี่มูลฐาน (Fundamental frequency) ความถี่ฟอร์แมนท์ (Formant frequency) และระดับพลังงาน (Energy profile) เป็นต้น ถึงแม้ว่าค่าลักษณะสำคัญแบบนี้จะมีประสิทธิภาพสูงในการรู้จำ แต่ยากในการสกัดจากสัญญาณ กลุ่มสุดท้ายเรียกว่าค่าลักษณะสำคัญแบบเอนเวโลปของสเปกตรัม (Spectral envelop feature) เป็นกลุ่มที่นิยมใช้กันมาก เนื่องจากค่าลักษณะสำคัญส่วนใหญ่สำหรับการรู้จำเสียงจะรวมอยู่ในข้อมูลเชิงสเปกตรัมนี้ อีกทั้งยังง่ายและสะดวกในการคำนวณหาค่าด้วย ตัวอย่างค่าลักษณะสำคัญแบบนี้ได้แก่ สัมประสิทธิ์การประมาณพันธะเชิงเส้น (Linear prediction coefficients: LPC), สัมประสิทธิ์เซปสตรัม (Cepstral coefficient) และพัฒนาการอีกมากมายจากเซปสตรัมปกติ อาทิเช่น สัมประสิทธิ์เซปสตรัมบนสเกลเมล (Mel frequency ceptral coefficients: MFCC) เซปสตรัมแบบหักลบค่าเฉลี่ย (Cepstral mean substraction: CMS) และเซปสตรัมแบบผ่านตัวกรองภายหลัง (Post filtered cepsturm: PFL) เป็นต้น นอกจากนั้น ยังมีการคำนวณค่าการเปลี่ยนแปลง (Derivative หรือ Delta) ของสัมประสิทธิ์เหล่านี้มาใช้เป็นค่าลักษณะสำคัญเพิ่มเติมได้ด้วย

สำหรับการคำนวณค่าลักษณะสำคัญแบบเอนเวโลปของสเปกตรัมจะมีขั้นตอนดังนี้

3.4.1 การเน้นสัญญาณขั้นต้น (Preemphasis)

เป็นขั้นตอนในการบีบอัดสัญญาณเสียง โดยนำสัญญาณเสียงผ่านตัวกรองลำดับหนึ่ง (First-order filter) ซึ่งจะเพิ่มอัตราส่วนสัญญาณต่อสัญญาณรบกวน (Signal to noise ratio)

3.4.2 การแบ่งเป็นส่วนย่อย (Frame)

เป็นขั้นตอนในการแบ่งสัญญาณเสียงเป็นส่วนย่อย ขนาดความยาวประมาณ 10 – 40 มิลลิวินาที ซึ่งทำให้สัญญาณเสียงมีคุณสมบัติเปลี่ยนแปลงตามเวลาน้อยมาก หรือไม่มีเลย เพื่อให้สามารถสร้างแบบจำลองการกระจายของหน่วยสัญญาณเสียงย่อยทางสถิติได้

3.4.3 การลดขอบด้วยฟังก์ชันหน้าต่าง(Smoothing window)

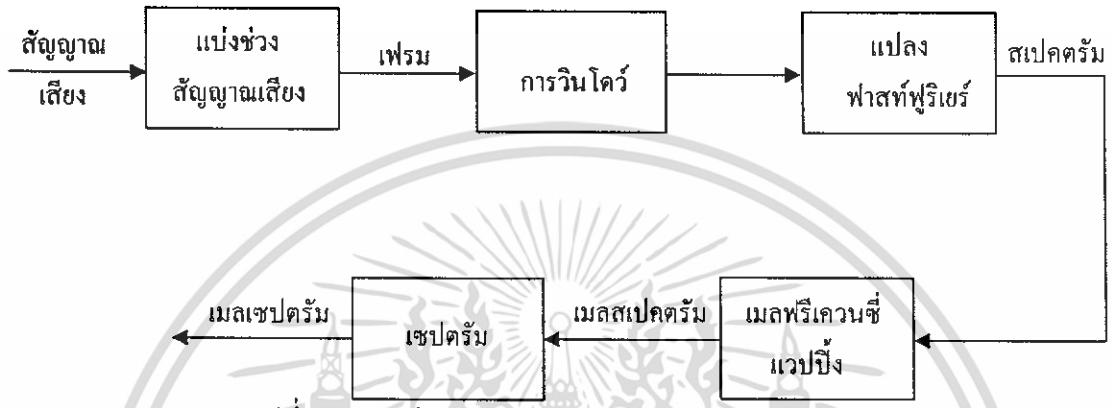
สำหรับปรับสัญญาณให้ราบเรียบ

3.4.4 การสกัดค่าลักษณะสำคัญ (Feature extraction)

ในส่วนนี้ จะทำการคำนวณค่าลักษณะสำคัญของสัญญาณเสียงในแต่ละส่วนย่อย ผลลัพธ์อยู่ในรูปแบบของเวกเตอร์ของค่าลักษณะสำคัญ (Feature vector) สำหรับแต่ละส่วนย่อย

3.4.5 การประมวลผลของเมลฟรีเควนซี่เซปตรัมโคเอฟฟีเซี่ยน (Mel-Frequency Cepstrum Coefficients Processor)

แผนภาพโครงสร้างของการประมวลผลเอ็มเอฟซีซีแสดงดังรูปที่ 3.6



รูปที่ 3.6 แสดงโครงสร้างของการประมวลผลเอ็มเอฟซีซี

โดยเราเลือกความถี่ในการสุ่ม 11025 เฮิร์ต ซึ่งความถี่นี้ครอบคลุมความถี่ของเสียงพูดมนุษย์ โดยจุดประสงค์ของการประมวลผลเอ็มเอฟซีซีคือการเลียนแบบพฤติกรรมของหูมนุษย์

3.4.5.1 การแบ่งช่วงสัญญาณ (Frame Blocking)

เสียงต้นฉบับจะถูกแบ่งเป็นบล็อก (block) แต่ละบล็อกประกอบด้วย 1024 แซมเปิลและบล็อกที่ติดกันมีช่วงที่ซ้อนทับกัน 512 แซมเปิล โดยบางแซมเปิลที่จุดสิ้นสุดของไฟล์ไม่ได้นำมาใช้ในการคำนวณ

3.4.5.2 การวินโดว์ (Windowing)

แต่ละบล็อกจะถูกวินโดว์เพื่อลดผลของการเปลี่ยนแปลงที่จุดเริ่มต้นและจุดสิ้นสุดของบล็อก เราเลือกใช้หน้าต่างแฮมมิ่ง (Hamming Window) ดังสมการ (3.1)

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad ;n = 0,1,2,...,N-1 \qquad (3.1)$$

โดยฟังก์ชั่นหน้าต่าง (window function) จะถูกคูณกับเมตริกซ์ (matrix) ที่ได้จากการแบ่งช่วงสัญญาณ

3.4.5.3 การแปลงฟาสท์ฟูริเยร์ (Fast Fourier Transform หรือ FFT)

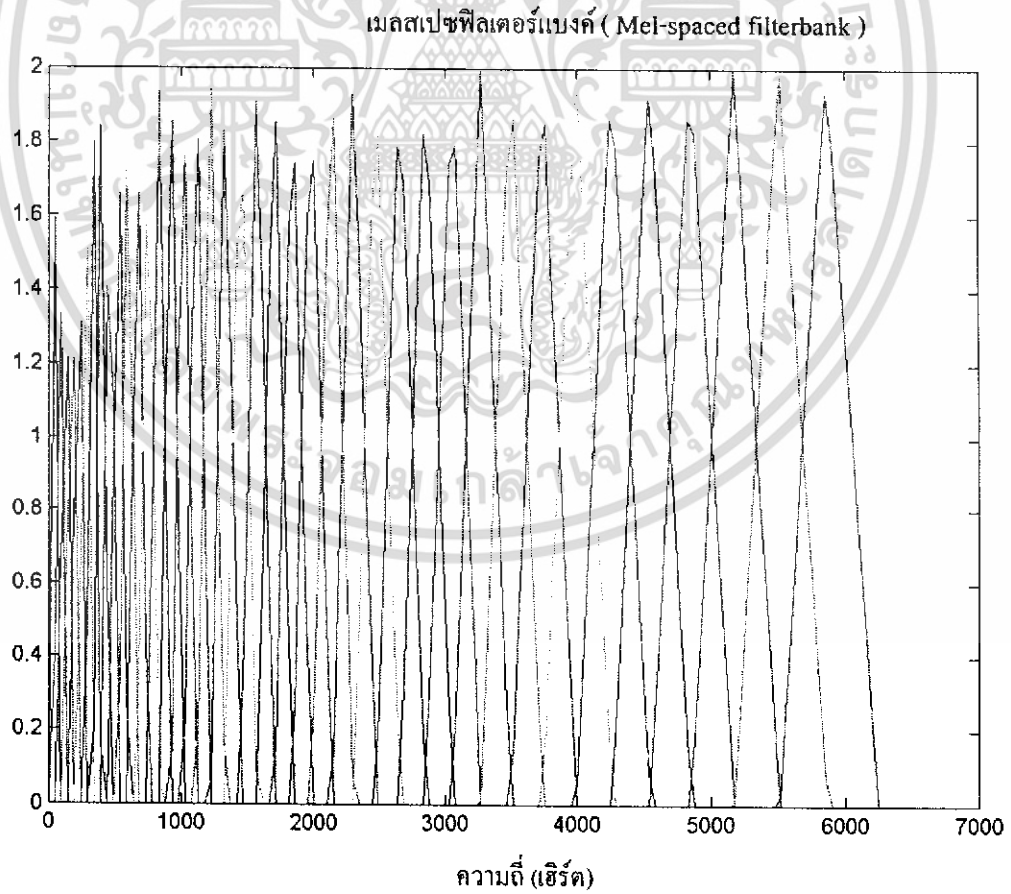
นำบล็อกที่ผ่านการวินโดว์แล้วมาผ่านการแปลงฟาสท์ฟูริเยร์ เพื่อแปลงข้อมูลจากโดเมนเวลาสู่โดเมนความถี่ และสร้างสัมประสิทธิ์สเปกตรัม

3.4.5.4 เมลฟรีเควนซี่แวปปิ้ง (Mel-Frequency Wrapping)

นำการแปลงเมลฟรีเควนซี่มาใช้ในการแปลงแต่ละสเปกตรัมให้อยู่ในรูปแบบเมลสเกล (melscale) โดยค่าที่อ้างอิงคือ ระดับเสียง 1 กิโลเฮิร์ตและ 40 เดซิเบลเหนือค่าขอบเขตการได้ยินของเพอเซปตรัม (perceptral) นิยามเป็น 1000 เมล (mel) สูตรตามสมการ (4.2)ใช้ในการคำนวณเมลจากความถี่ f ที่ได้

$$mel(f) = 2595 * \log_{10}(1 + f/700) \qquad (3.2)$$

สเปกตรัมสามารถสร้าง (simulate) โดยใช้ฟิลเตอร์แบงค์ที่มีการกระจายตัวด้วยระยะห่างที่สม่ำเสมอบนเมลสเกล ฟิลเตอร์แบงค์ตอบสนองต่อการแบนด์พาสความถี่เป็นรูปสามเหลี่ยม โดยระยะห่างและแบนด์วิดธ์จะถูกกำหนดโดยค่าความแตกต่างระหว่างค่าคงที่ของความถี่เมล เราเลือกฟิลเตอร์แบงค์ขนาด 40 ฟิลเตอร์แบงค์ที่เราสร้างขึ้นเป็นดังรูปที่ 3.7



รูปที่ 3.7 แสดงเมลสเปซฟิลเตอร์แบงค์ขนาด 40

3.4.5.5 เซปตรัม (Cepstrum)

ใช้ฟังก์ชันล็อก (Log) เพื่อแปลงเมลสเปกตรัมกลับสู่โดเมนเวลา เพื่อหาเมลฟรีเควนซี่เซปตรัมโคเอฟฟีเชี่ยนเนื่องจากเมลสเปกตรัมโคเอฟฟีเชี่ยนเป็นจำนวนจริง เราจึงใช้การแปลงดิสครีสโคซายน์ (Discrete Cosine Transform หรือ DCT) โดยเอ็มเอฟซีซีคำนวณได้จากสมการ (3.3)

$$\widetilde{c}_n = \sum_{k=1}^{K} (\log \widetilde{S}_k) \cos\left[n\left(k - \frac{1}{2} \right) \frac{\pi}{K} \right] \qquad ; n = 1,2,...,K \qquad (3.3)$$

ซึ่ง $\widetilde{S}_k, k = 1,2,...,K$ แทนสัมประสิทธิ์เมลสเปกตรัม

เราจะได้เมลฟรีเควนซี่เซปตรัมโคเอฟฟีเชี่ยนสำหรับเสียงในแต่ละช่วงสัญญาณซึ่งเราเรียกว่า อคูสติกเวคเตอร์ (acoustic vector)ในแถวแรกของเมตริกซ์ที่ได้เราไม่นำมาใช้ เพราะเป็นค่าเฉลี่ยของสัญญาณอินพุทซึ่งเป็นข้อมูลที่มีความสำคัญน้อยมาก ทำให้เราได้เมตริกซ์ขนาด 39 แถว

3.5 การรู้จำ (Recognition)

ขั้นตอนนี้ประกอบด้วย 2 หน้าที่หลัก คือการนำเวกเตอร์ของค่าลักษณะสำคัญของสัญญาณเสียง ที่อยู่ในชุดอ้างอิงหรือชุดฝึกฝน มาทำการเรียนรู้ เมื่อเรียนรู้แล้วเวกเตอร์ของสัญญาณเสียงที่ต้องการทดสอบการรู้จำ จะถูกนำเข้ามาเทียบเคียงเพื่อรู้จำ ขั้นตอนในการเรียนรู้นั้นขึ้นอยู่กับวิธีในการรู้จำของระบบนั้นๆ บางวิธีก็เพียงแค่เก็บข้อมูลชุดเรียนรู้ไว้เปรียบเทียบกับข้อมูลชุดทอดสอบเท่านั้น เช่น วิธีการรู้จำแบบหาค่าระยะห่างยูคลิเดียน (Euclidean distance) วิธีไดนามิกไทม์วาร์ปปิง (Dynamic time warping: DTW) เป็นต้น ในขณะที่บางวิธี จะนำข้อมูลชุดเรียนรู้ไปแปลงเป็นค่าอ้างอิงที่ต้องการ เช่น โครงข่ายประสาทเทียม (Artificial neural networks: ANN) จะนำข้อมูลชุดเรียนรู้ไปผ่านโครงข่ายที่สร้างขึ้น เพื่อจดจำรูปแบบ และเก็บเป็นค่าน้ำหนัก (Weight) แทน วิธีควอนไตซ์แบบเวกเตอร์ (Vector quantization: VQ) ซึ่งจะแทนเวกเตอร์ทั้งหมด ของแต่ละสัญญาณเสียงอ้างอิงด้วยเวกเตอร์จำนวนไม่มาก หรือการใช้แบบจำลองฮิดเดนมาร์คอฟ (Hidden markov model: HMM) โดยนำข้อมูลชุดฝึกฝนไปผ่านแบบจำลองที่สร้างขึ้นเพื่อจดจำรูปแบบ และเก็บค่าทางสถิติและค่าความน่าจะเป็นของแต่ละสถานะไว้ เป็นต้น แต่ทั้งหมดจะมีพื้นฐานอยู่ที่การคำนวณระยะห่างของรูปแบบที่จะรู้จำและนำค่าระยะห่างที่ได้ไปใช้รู้จำตามแต่ละวิธีนั้นๆ

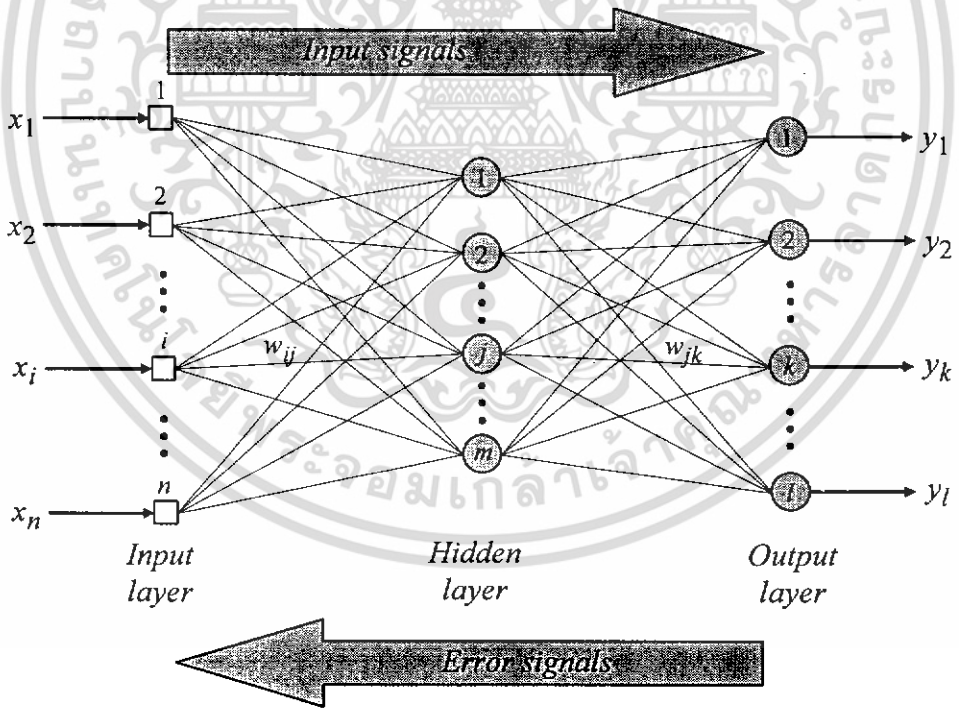
การเลือกใช้วิธีการรู้จำ ขึ้นอยู่กับข้อกำหนดของงาน เช่น วิธี DTW และ ANN เหมาะสมกับระบบแบบกำหนดคำพูดตายตัว ในขณะที่วิธี VQ และ HMM จะเหมาะสมกับระบบงานที่เป็นแบบไม่กำหนดคำพูดมากกว่า อีกทั้งยังต้องพิจารณาข้อดี-ข้อเสียของแต่ละวิธีด้วย

ในปริญญานิพนธ์ ได้ทำการออกแบบส่วนรู้จำโดยใช้วิธีนิวรอลเน็ตเวิร์คแบบแบ็คพรอพพาเกชั่น (Backpropagation) ในการตัดสินใจเสียงที่เข้ามาเป็นเสียงของคำสั่ง 10 คำสั่ง ขั้นตอนของการฝึกสอน จะใช้เสียงในการฝึกสอนของผู้พูด 20 คน เสียงละ 1 ครั้ง 10 เสียง รวม 200 เสียง โดยจะใช้รายละเอียดโมดูลของนิวรอลเน็ตเวิร์ค ดังนี้

INPUT NEURONS	=	624
HIDDEN NEURONS	=	20
OUTPUT NEURONS	=	10
LEARNING RATE	=	0.1

3.5.1 กระบวนการ Back – propagation โดยใช้ Neural network

เป็นการเรียนรู้ค่าพารามิเตอร์ของอินพุตแต่ละตัวโดยใช้นิวรอลเน็ตเวิร์ก โดยเปรียบเทียบกับรูปแบบเอาท์พุตที่กำหนดตายตัว ระหว่างที่ค่าที่ได้จากการเรียนค่าพารามิเตอร์ของอินพุตยังไม่ตรงกับรูปแบบเอาท์พุตที่ต้องการ ระบบจะมีการปรับค่าน้ำหนักในนิวรอลเนตเวิร์กเพื่อให้ได้ค่าที่ตรงกับรูปแบบเอาท์พุตที่ต้องการมากที่สุด



รูปที่ 3.8 แสดง Back – propagation ขนาด 3 เลเยอร์

3.5.1.1 การกำหนดค่าเริ่มต้น

ทำการกำหนดค่าน้ำหนักและค่าเทรโชลด์ในเนตเวิร์ก โดยสุ่มความถี่ค่าให้กระในช่วงตามสมการข้างล่าง

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right) \qquad (3.4)$$

โดย Fi คือจำนวนอินพุตทั้งหมดในเนตเวิร์ก

3.5.1.2 การคำนวณค่า

คำนวณค่าในระบบ back – propagation โดยค่าอินพุต $x1(p), x2(p),\ldots, xn(p)$ จะได้ค่าเป็น $yd,1(p), yd,2(p),\ldots, yd,n(p)$ ที่เลเยอร์ถัดไป

(1) คำนวณค่าเอาท์พุตที่ได้จริงใน hidden layer ดังสมการ

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p)\cdot w_{ij}(p) - \theta_j\right] \qquad (3.5)$$

โดย n คือ จำนวนอินพุตทั้งหมดของนิวรอลใน hidden layer โดยใช้สมการของซิกมอยด์

(2) คำนวณค่าเอาท์พุตที่ได้จริงใน output layer ดังสมการ

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p)\cdot w_{jk}(p) - \theta_k\right] \qquad (3.6)$$

โดย m คือ จำนวนอินพุตนิวรอลใน output layer

3.5.1.3 การปรับปรุงค่าน้ำหนัก

การปรับปรุงค่าน้ำหนักในนิวรอลเนตเวิร์กจะมีความสัมพันธ์กับค่าความแตกต่างของเอาท์พุตโดยแบ่งเป็น

(1) คำนวณค่าควบคุมให้น้ำหนักเปลี่ยนแปลงเข้าสู่จุดดีที่สุด ใน output layer จากสมการ

$$\delta_k(p) = y_k(p)\cdot\left[1 - y_k(p)\right]\cdot e_k(p) \qquad (3.7)$$

โดย $$e_k(p) = y_{d,k}(p) - y_k(p) \qquad (3.8)$$

ต่อจากนั้น คำนวณค่าน้ำหนักที่ถูกต้อง จากสมการ

$$\Delta w_{jk}(p) = \alpha\cdot y_j(p)\cdot\delta_k(p) \qquad (3.9)$$

ปรับปรุงค่าน้ำหนักที่ควรจะเป็น จากสมการ

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \qquad (3.10)$$

(2) คำนวณค่าควบคุมให้น้ำหนักเปลี่ยนแปลงเข้าสู่จุดดีที่สุด ใน hidden layer จากสมการ

$$\delta_j(p) = y_j(p)\cdot[1-y_j(p)]\cdot\sum_{k=1}^{l}\delta_k(p)\,w_{jk}(p) \qquad (3.11)$$

ต่อจากนั้น คำนวณค่าน้ำหนักที่ถูกต้อง จากสมการ

$$\Delta w_{ij}(p) = \alpha\cdot x_i(p)\cdot\delta_j(p) \qquad (3.12)$$

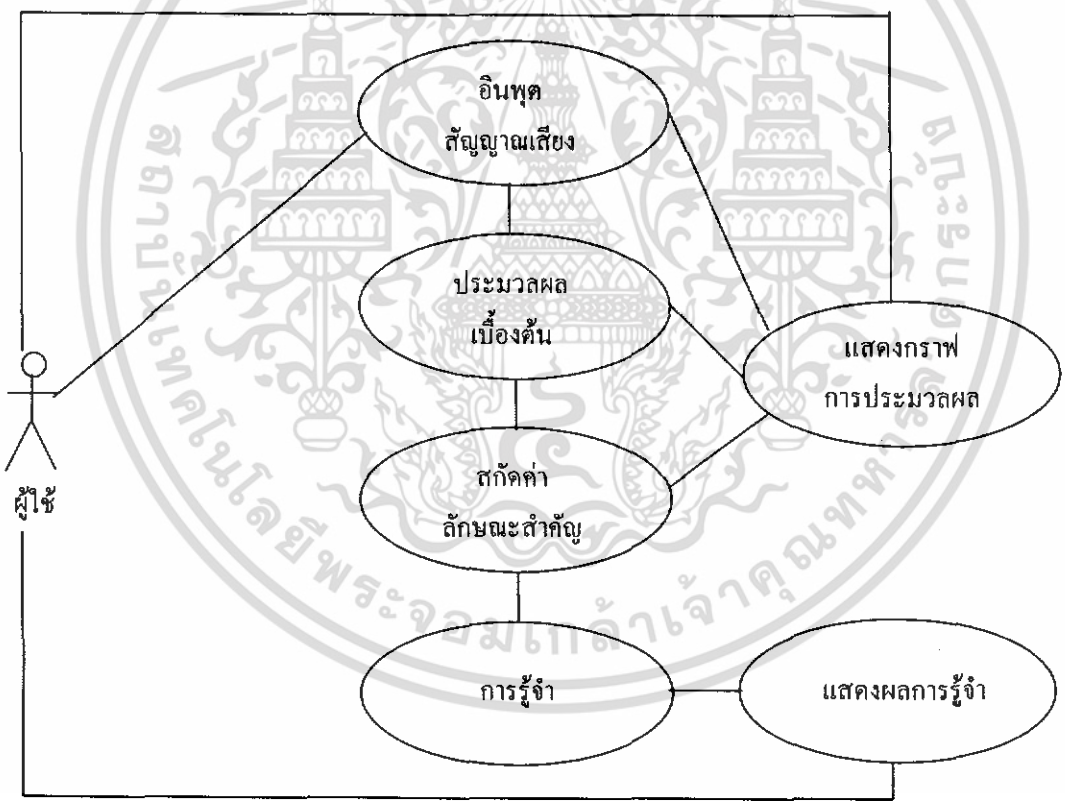
ปรับปรุงค่าน้ำหนักที่ควรจะเป็น จากสมการ

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \qquad (3.13)$$

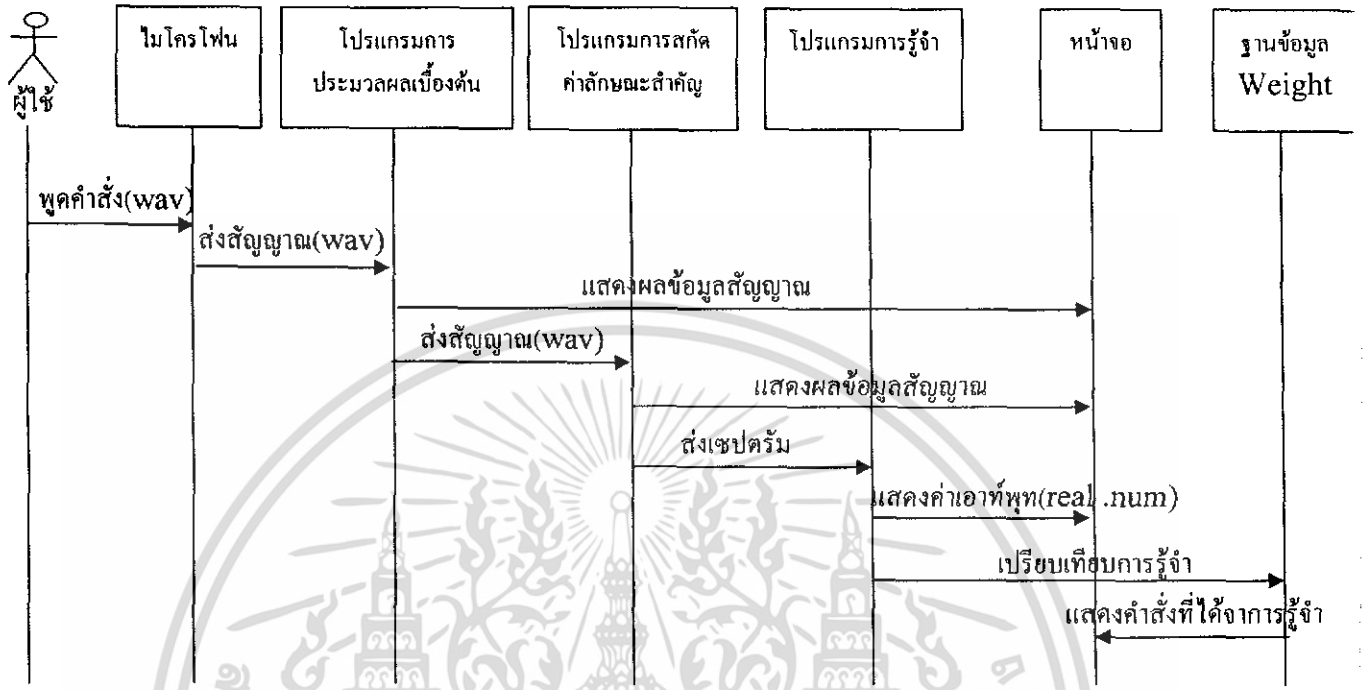
3.5.1.3 การทำซ้ำ

จากจากเสร็จสิ้นการปรุงปรุงค่าน้ำหนักทั้งหมด ระบบจะกลับไปคำนวณที่ขั้น 3.5.1.2 โดยจะทำซ้ำไปเรื่อยจนกระทั่งค่า error ที่ได้จะเข้ามาสู่จุดที่เราพอใจ

3.6 UML Diagram ระบบรู้จำเสียง



3.7 Sequence Diagram



บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลอง

ระบบจะทำการบันทึกเสียงคำสั่งทุกเสียง ด้วยความละเอียดขนาด 16 บิท และความถี่ในการสุ่ม 11025 เฮิร์ต เนื่องจากเป็นช่วงความถี่ที่เหมาะสมในการรู้จำเสียงคำสั่งของโครงงานนี้ โดยทางคณะผู้จัดทำแบ่งการทดลองเป็น 2 การทดลอง คือ

1. การทดลองเพื่อหาช่วงการตัดเสียงที่เหมาะสมในการรู้จำเสียง ใช้ผู้ทดลอง 2 คน ผู้ชาย 2 คน โดยทดลองในห้องเงียบ
2. การทดลองเพื่อหาจำนวนรูปแบบอ้างอิงที่เหมาะสมสำหรับการรู้จำเสียง ใช้ผู้ทดลองคนเดียว โดยทดลองในห้องเงียบ
3. การทดลองเพื่อวิเคราะห์สภาวะแวดล้อมที่เหมาะสมกับการรู้จำเสียง ใช้ผู้ทดลองคนเดียว เป็นผู้ชาย โดยทดลองในสภาวะแวดล้อมที่ต่างกัน 4 กรณี คือ
   กรณีที่ 1 ทดลองในห้องเงียบ
   กรณีที่ 2 ทดลองในห้องที่มีเสียงรบกวนจากห้องทำงาน

4.2 การสร้างรูปแบบอ้างอิง

ผู้ทดลองแต่ละคนทำตามขั้นตอนดังนี้

1. บันทึกเสียงคำสั่งทุกเสียง (บันทึกครั้งแรก) ที่ต้องการให้ระบบเรียนรู้

2. นำข้อมูลเสียงทุกคำสั่งทำการกรองเสียงและตัดหัวท้ายเสียงโดยหาค่า amplitude ที่มากที่สุดของเสียงนั้นๆ และกำหนดจุดเริ่มต้นของเสียงโดยดูจากกำหนดค่าเริ่มต้นมีค่าเป็น 0.2 เท่าของค่าสูงสุด หากจุดใดมีค่าเกินค่าเริ่มต้นก่อน จุดนั้นจะกลายเป็นจุดเริ่มต้น และจุดสุดท้ายจะห่างจากจุดเริ่มต้นเท่ากับ 10,000 แซมเปิ้ล ซึ่งทุกๆที่ถูกตัดหัวและท้ายเสียงจะมีความยาวของเสียงเท่ากัน

3. นำข้อมูลเสียงของแต่ละคำสั่งที่ทำการกรองเสียงและตัดหัวท้ายเสียงมาทำการสกัดค่าของเสียงโดยใช้วิธีเมลฟรีเควนซี่เซปตรัมโคเอฟฟีเชี่ยน (MFCC) โดยเสียงทุกเสียงจะถูกเก็บไว้ในรูปแบบไฟล์ (.txt)

4. นำค่าพารามิเตอร์ของทุกๆเสียง มาเก็บไว้ที่ไฟล์เดียวเพื่อเป็นฐานข้อมูลของทุกอินพุต (ในการทดลองใช้ผู้ทดลองทั้งหมด 20 คน โดยแต่ละคนจะพูดคำสั่ง 10 คำสั่งๆ ละ 2 ครั้ง ที่แตกต่างกัน รวมเสียงที่ใช้เป็นฐานข้อมูลทั้งหมด 200 เสียง)

5. นำค่าพารามิเตอร์ทั้งหมดของทุกเสียง มาทำการเทรนนิ่ง ด้วยนิวรอลเน็ตเวิร์ก แบบ Back – Propagation ซึ่งแต่ละสียงจะมี feature ทั้งหมด 624 feature ซึ่งถือเป็นจำนวนนิวรอลของ input layer และกำหนดจำนวนนิวรอลใน hidden layer มีจำนวนเท่ากับ 20 โหนด ในส่วนของ

output layer จะเก็บ feature ของ output แต่ละเสียงไว้ซึ่งมีทั้งหมด 10 feature และเท่ากับจำนวนนิวรอลในชั้น output layer จากนั้นในการเทรนแต่ละรอบ ค่า weight ทุกอันระหว่างนิวรอลในแต่ละเลเยอร์ จะมีการปรับปรุงค่าเพื่อให้ตรงกับ feature ของเอาท์พุตที่เราต้องการ จนกว่าจะได้ค่า weight ที่เหมาะสม

4.3 การทดสอบ

ผู้ทดลองแต่ละคนทำตามขั้นตอนดังนี้

1. นำคำสั่งเสียงที่ต้องการทดสอบเพียง 1 คำสั่ง ผ่านทางไมโครโฟน

2. คำสั่งเสียง 1 คำสั่งจะถูกนำมาผ่านขั้น Pre-Processing ซึ่งจะได้ข้อมูลเสียงที่ผ่านการกรองความถี่และตัดหัวท้ายเสียง มีความยาวเท่ากับ 10,000 แซมเปิ้ล

3. นำข้อมูลเสียงที่ผ่านกระบวนการข้างต้นทากรสกัดค่าผ่านขั้นตอนเมลฟรีเควนซี่เซปตรัมโคเอฟฟิเชี่ยนจะได้พารามิเตอร์ของคำสั่งนั้น (1 คำสั่ง จะมีทั้งหมด 624 พารามิเตอร์)

4. นำพารามิเตอร์ทั้งหมดของเสียงนั้นผ่านกระบวนการ Back – Propagation ซึ่งแต่ละพารามิเตอร์ของเสียงถูกนั้น ไปคูณกับค่าที่เชื่อมกันระหว่างแต่ละนิวรอล จนได้ค่าfeature ที่แสดงออกมาที่ output layer

5. นำค่า feature ที่แสดงออกมาที่ output layer เปรียบเทียบดูว่าตรงกับเสียงใด

4. นำคำสั่งนั้นไปใช้ในการสั่งงาน

โดยคำสั่ง 10 คำสั่งที่ใช้ทดสอบมีดังนี้

คำสั่งที่	คำพูด
1	Word
2	Power Point
3	Excel
4	Note Pad
5	Paint
6	Internet
7	E-mail
8	MSN
9	Music
10	Pinball

ตารางที่ 4.1 แสดงคำสั่ง 10 คำสั่งที่ใช้ทดสอบการรู้จำเสียงคำสั่ง

4.4 ผลการทดสอบ

4.4.1การทดลองเพื่อหาช่วงการตัดเสียงที่เหมาะสมในการรู้จำเสียง

4.4.1.1 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 1

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	Endpoint = 0.1*max		Endpoint = 0.2*max		Endpoint = 0.3*max	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	1	1	1	1
2	10	2	2	10	10	10
3	3	3	7	7	7	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	3	6	6	6	7	7
7	7	7	7	7	7	7
8	3	7	7	7	7	7
9	9	9	9	9	9	9
10	10	2	10	10	10	10
เปอร์เซ็นต์ความถูกต้อง	70%	80%	80%	80%	60%	70%

ตารางที่ 4.2 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 1

4.4.1.2 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 2

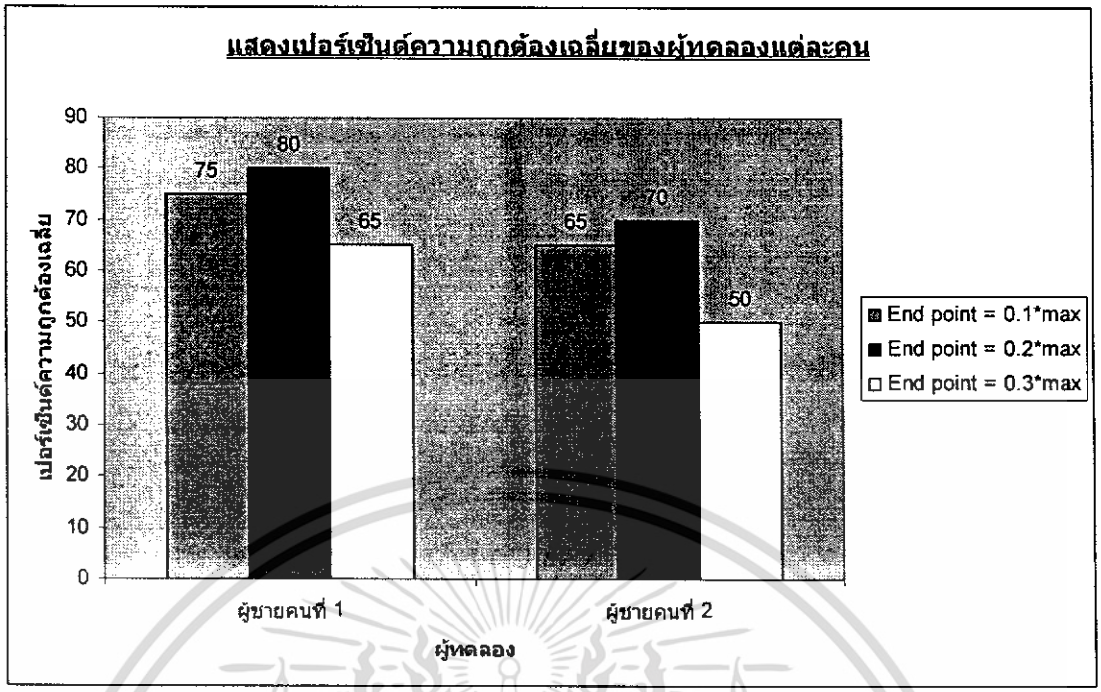
คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	Endpoint = 0.1*max		Endpoint = 0.2*max		Endpoint = 0.3*max	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	1	1	1	1
2	2	2	5	2	5	10
3	7	7	7	7	7	7
4	4	4	4	4	4	4
5	5	5	5	2	5	5
6	7	7	7	6	7	6
7	7	7	7	7	7	7
8	7	7	8	7	7	7
9	9	9	9	9	9	9
10	10	2	10	10	5	10
เปอร์เซ็นต์ความถูกต้อง	70%	60%	70%	70%	40%	60%

ตารางที่ 4.3 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ของผู้ทดลองผู้ชายคนที่ 2

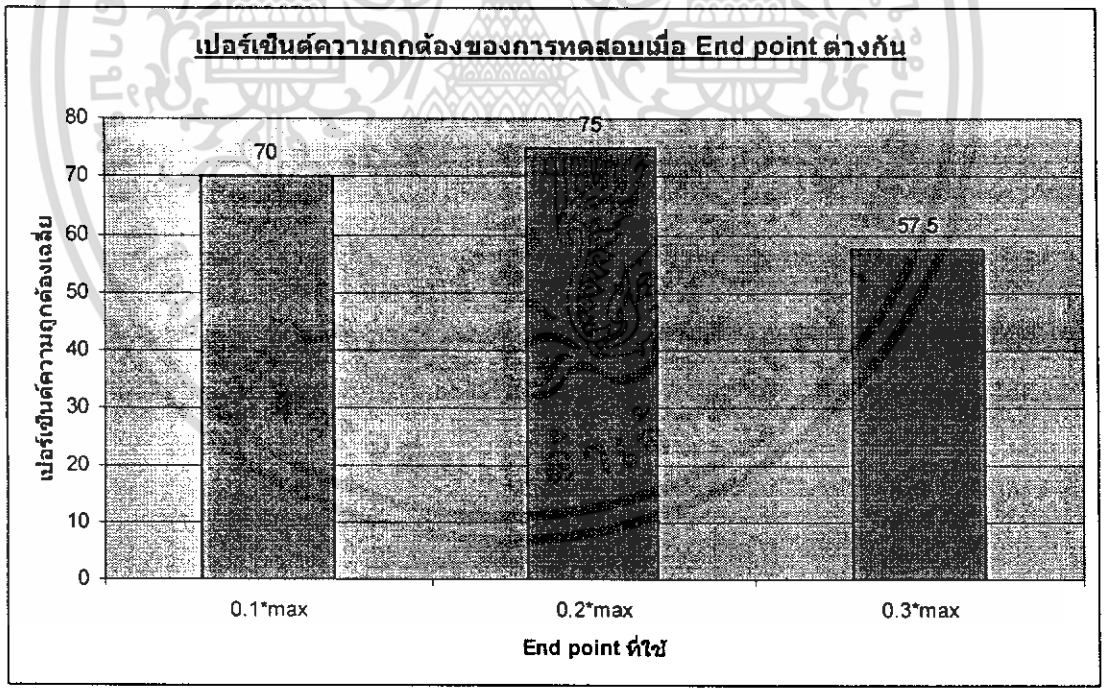
4.4.1.3 ผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ ที่จุดตัดเสียงต่างๆ

ผู้ทดลอง	เปอร์เซ็นต์ความถูกต้องเฉลี่ย		
	Endpoint = 0.1*max	Endpoint = 0.2*max	Endpoint = 0.3*max
ผู้ชายคนที่ 1	75%	80%	65%
ผู้ชายคนที่ 2	65%	70%	50%
เปอร์เซ็นต์ความถูกต้องเฉลี่ยรวม	70%	75%	57.5%

ตารางที่ 4.4 แสดงผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในห้องเงียบ



รูปที่ 4.1 กราฟแสดงเปอร์เซ็นต์ความถูกต้องเฉลี่ยของผู้ทดลองแต่ละคน เมื่อสั่งงานในห้องเงียบ



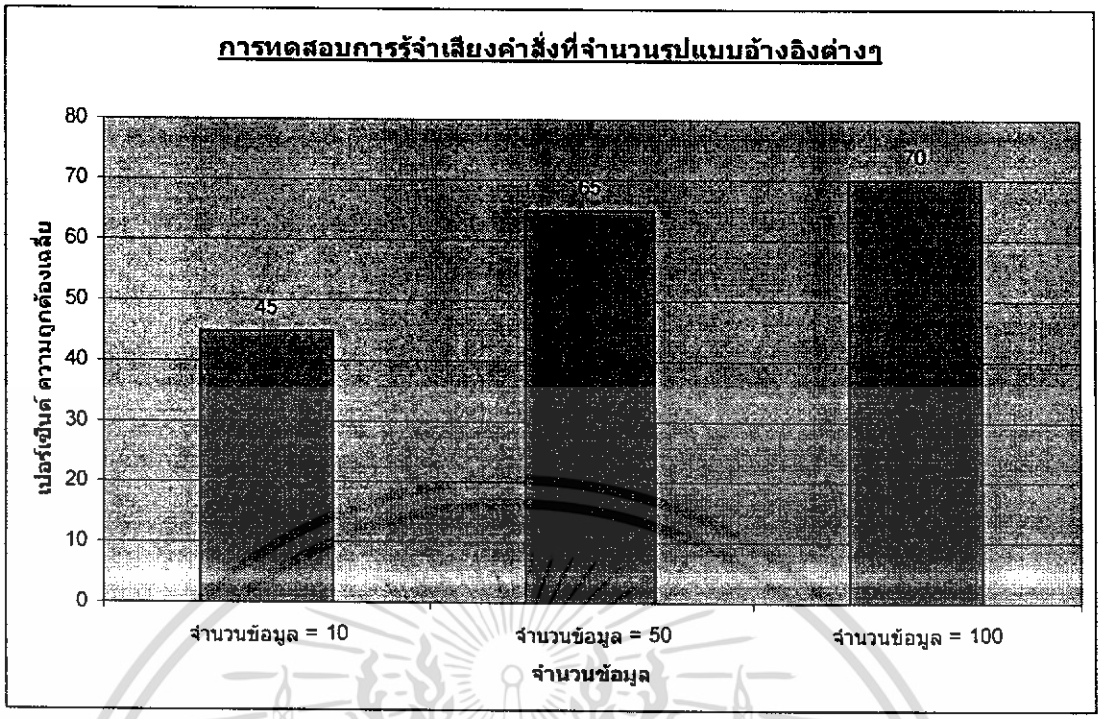
รูปที่ 4.2 กราฟแสดงผลการเปรียบเทียบเปอร์เซ็นต์ความถูกต้องของการทดสอบการรู้จำเสียงคำสั่งที่ใช้ขนาดบล็อกต่างกัน เมื่อสั่งงานในห้องเงียบ

4.4.2 การทดลองเพื่อหาจำนวนรูปแบบอ้างอิงที่เหมาะสมสำหรับการรู้จำเสียง

4.4.2.1 ผลการทดสอบการรู้จำเสียงคำสั่งที่จำนวนรูปแบบอ้างอิงต่างๆ ของผู้ทดลองผู้ชายคนที่ 1 โดยใช้ Endpoint = 0.2 * max

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง					
	จำนวนข้อมูล = 10		จำนวนข้อมูล = 50		จำนวนข้อมูล = 100	
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 1	ครั้งที่ 2
1	1	1	1	1	1	1
2	5	5	5	2	5	2
3	7	7	7	7	3	7
4	4	4	4	4	4	4
5	5	5	5	2	5	5
6	7	7	7	6	7	6
7	7	7	7	7	7	7
8	7	7	8	7	8	7
9	9	9	9	9	9	9
10	5	5	5	10	10	10
เปอร์เซ็นต์ความถูกต้อง	50%	40%	60%	70%	70%	70%

ตารางที่ 4.5 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานจำนวนรูปแบบอ้างอิงที่ต่างกันของผู้ทดลองผู้ชายคนที่ 1 โดยใช้ Endpoint = 0.2 * max



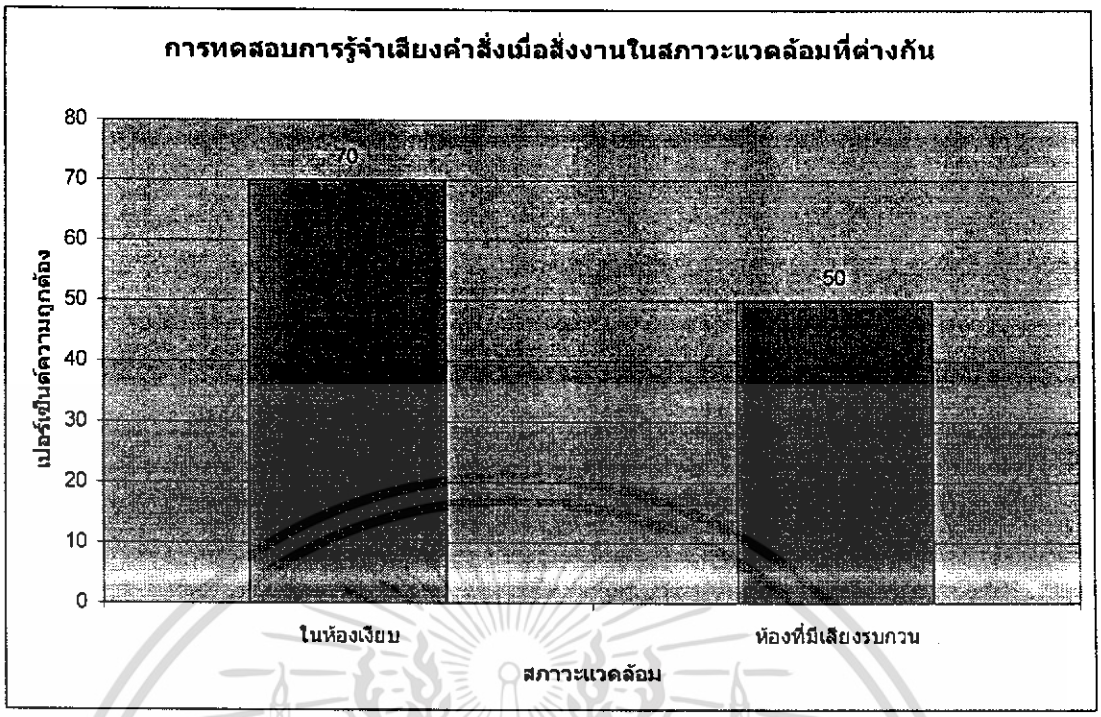
รูปที่ 4.3 กราฟแสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในรูปแบบอ้างอิงที่ต่างกัน โดยใช้ Endpoint = 0.2*max

4.4.3 การทดลองเพื่อวิเคราะห์สภาวะแวดล้อมที่เหมาะสมกับการรู้จำเสียง

4.4.3.1 ผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน ของผู้ทดลองผู้ชายคนที่ 1 โดยใช้ Endpoint = 0.2 * max

คำสั่งที่ต้องการทดสอบคำสั่งที่	ผลการรู้จำเสียงคำสั่ง	
	ในห้องเงียบ	ห้องที่มีเสียงรบกวน
1	1	1
2	5	10
3	3	7
4	4	4
5	5	10
6	7	7
7	7	7
8	7	7
9	9	9
10	10	10
เปอร์เซ็นต์ความถูกต้อง	70%	50%

ตารางที่ 4.6 แสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน ของผู้ทดลองผู้ชายคนที่ 1 โดยใช้บล็อกขนาด 1024



รูปที่ 4.4 กราฟแสดงผลการทดสอบการรู้จำเสียงคำสั่งเมื่อสั่งงานในสภาวะแวดล้อมที่ต่างกัน โดยใช้ Endpoint = 0.2*max

บทที่ 5
บทสรุป

5.1 บทนำ

โดยส่วนใหญ่จะพัฒนางานแบบการตรวจสอบผู้พูด (Speaker Verification) มากกว่าที่จะเป็นการระบุผู้พูด (Speaker Identification) เพราะนำไปใช้ได้หลากหลายกว่า โดยผู้ใช้งานในแบบแรกมักจะให้ข้อมูลอื่น ๆ เพิ่มเติมด้วย ได้แก่ชื่อหรือรหัสบัตรประจำตัวเพื่อเป็นข้อมูลประกอบ สำหรับอินพุตเสียงภาษาอังกฤษมีการพัฒนาระบบรักษาความปลอดภัยที่ใช้ในการระบุผู้พูดจำหน่ายในท้องตลาดแล้ว ตัวอย่างผู้จำหน่ายระบบรักษาความปลอดภัยซึ่งใช้เสียงพูดเป็นอินพุต ได้แก่ Moscom, Sensory, Sonetech, และ Veritel ซึ่งถ้าจะนำมาใช้กับเมืองไทย จะพบปัญหาเรื่องความแตกต่างของลักษณะเสียงของภาษาไทยจากเสียงภาษาอังกฤษ ซึ่งจะต้องมีการปรับปรุงเทคนิคต่างๆ เพื่อให้เหมาะสมกับผู้ใช้เสียงภาษาไทย ผู้จัดทำจึงมีแนวคิดเกี่ยวกับการทำวิจัยในโครงการระบุผู้พูดด้วยเสียงอินพุตภาษาไทยขึ้น ซึ่งเป็นหัวข้อที่ค่อนข้างใหม่ทีเดียวสำหรับวงการงานวิจัยในเมืองไทย

ระบบรู้จำเสียงมีความสำคัญมากกับการเพิ่มประสิทธิภาพของระบบรักษาความปลอดภัย และยังเป็นพื้นฐานที่สำคัญในการพัฒนาระบบรู้จำเสียงพูด (Speech recognition system) สำหรับภาษาไทยอีกด้วย ในระหว่างการศึกษาผู้จัดทำได้ทำการคิดค้นและออกแบบระบบต้นแบบสำหรับภาษาไทยที่ให้อัตราความถูกต้องเฉลี่ยสูงที่สุด กับผู้พูดจำนวน 10 - 20 คน และจะพัฒนาระบบนี้ให้สามารถทำงานได้ดีกับผู้พูดจำนวนมากยิ่งขึ้นและจะพัฒนาระบบระบุผู้พูดที่ใช้งานได้กับโปรแกรมต่างๆในระบบคอมพิวเตอร์ และคิดค้นเทคนิคใหม่ๆ เพื่อระบบรู้จำเสียงที่ดีขึ้นอีกด้วย

คุณภาพของระบบรู้จำจะแตกต่างกันไป ขึ้นอยู่กับปัจจัยต่างๆ ได้แก่

1. รายละเอียดของวิธีการอัดเสียงพร้อมทั้งอุปกรณ์ที่เกี่ยวข้อง
2. เทคนิคการประมวลผลสัญญาณเสียงเบื้องต้น
3. เทคนิคการสกัดค่าลักษณะสำคัญ
4. เทคนิคการรู้จำ
5. ปริมาณผู้พูด

5.2 สรุปผลการทดลอง

จากการทดลองระบบการรู้จำเสียงคำสั่งจำนวน 10 คำสั่ง โดยใช้สัญญาณเสียงด้วยความละเอียดขนาด 16 บิท และความถี่ในการสุ่ม 11025 เฮิร์ต ผลการทดลองสรุปได้ดังนี้

1. ค่าจุดตัดเสียงและจำนวนรูปแบบอ้างอิงมีผลต่อระบบรู้จำเสียง ดังนั้นจุดตัดเสียงที่เหมาะสม คือ 0.2 * max ส่วนจำนวนรูปแบบอ้างอิงยิ่งมีจำนวนมากยิ่งมีประสิทธิภาพในการรู้จำมาก แต่ถ้ามีมากเกินไปจะใช้เวลาในการเรียนรู้นาน ดังนั่นจึงเลือกจำนวนที่ใช้ คือ 100 เสียง ซึ่งมีเปอร์เซ็นต์ความถูกต้องสูงที่สุดคือ 70 %
2. เสียงพูดของคน ๆ เดียวกันในแต่ละคำสั่ง ในการพูดแต่ละครั้งพบว่าเสียงที่พูดออกมาไม่เหมือนกัน และลักษณะของคำสั่งที่มีการออกเสียงคล้ายกัน มีผลทำให้เปอร์เซ็นต์ความถูกต้องลดลง
3. การบันทึกเสียงเพื่อนำมาสร้างรูปแบบอ้างอิง และเสียงที่ต้องการทดสอบ ควรจะบันทึกในสภาพห้องที่เงียบ มีเสียงรบกวนน้อย และมีอุปกรณ์บันทึกเสียงที่ดี เพื่อให้ได้สัญญาณเสียงที่มีคุณภาพ
4. ในการรู้จำเสียงโดยทำการหาค่าพารามิเตอร์ที่เป็นตัวแทนเสียงด้วยวิธีเมลฟรีเควนซี่เซปตรัม โคเอฟฟิเซี่ยน และนำพารามิเตอร์เสียงทั้งหมดผ่านกระบวนการรู้จำ โดยวิธี Back – Propagation โดยใช้ Neural network สามารถรู้จำเสียงได้ดีระดับหนึ่ง โดยมีเปอร์เซ็นต์ความถูกต้องเท่ากับ 70.00%

5.3 บทวิจารณ์และแนวทางพัฒนา

การที่ผลการทดลองออกมายังไม่สามารถรู้จำเสียงได้ดีเท่าที่ควร เนื่องจากเกิดสัญญาณรบกวนในขณะทำการอัดเสียง และลักษณะของคำสั่งที่มีการออกเสียงคล้ายกัน ทำให้ค่าพารามิเตอร์ที่ได้จากการสกัดค่าไม่แตกต่างกันเท่าที่ควรจนทำให้ผลการทดสอบการรู้จำเกิดความคลาดเคลื่อน

นอกจากนั้น ระยะเวลาในการเรียนรู้ค่านำหนักในนิวรอลเน็ตเวิร์กต้องใช้ระยะเวลานาน โดยทางผู้จัดทำไม่มีเวลาเพียงพอในการทำการเรียนรู้ค่าน้ำหนัก จึงทำให้ประสิทธิภาพในการรู้จำต่ำลงได้เช่นเดียวกัน

การพัฒนาและปรับปรุงในขั้นต่อไป ควรปรับปรุงวิธีหาค่าพารามิเตอร์ที่เป็นตัวแทนเสียงให้เหมาะสม หรืออาจจะปรับเปลี่ยนวิธีการสกัดค่าเสียงให้มีค่าที่ต่างกัน ซึ่งคาดว่าจะทำให้ประสิทธิภาพในการรู้จำเสียงดีขึ้น

บรรณานุกรม

[1] นาย รวินทร์ วิรัชพินทุ และ นางสาวสุวารี เหลือเพิ่มพร "Voice-to-Command Agent" ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต ภาควิชา คอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2546

[2] รศ.ดร.มนัส สังวรศิลป์ และวรรัตน์ ภัทรอมรกุล , คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์ , อินโฟเพรส , 2543

[3] ดร.จุฬารัตน์ ตันประเสริฐ "การสร้างรหัสลับด้วยเสียงพูดภาษาไทย" [Online]. Available :http://micro.se-ed.com/content/mc188/MC188_75.asp.

[4] ชัย วุฒิวิวัฒน์ชัย, สุทัศน์ แซ่ตั้ง และวารินทร์ อัจฉริยะกุลพร "ความก้าวหน้าของการพัฒนาระบบระบุผู้พูดภาษาไทย" ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ

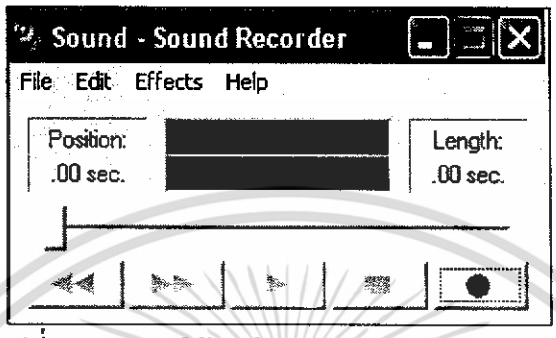
[5] Chenwithisuk, K. 1995. Thai Printed Characters Recognition Using A Neural Network And The Syntatic Method. M.S. thesis, Chulalongkorn Univ., Bangkok.

[6] Sriviroolchai, T. 1998. Thai Printed Characters Recognition Using Principal Component Analysis And Neural Networks. M.S. thesis, Chulalongkorn Univ., Bangkok.

ภาคผนวก ก
การสร้างรูปแบบอ้างอิง

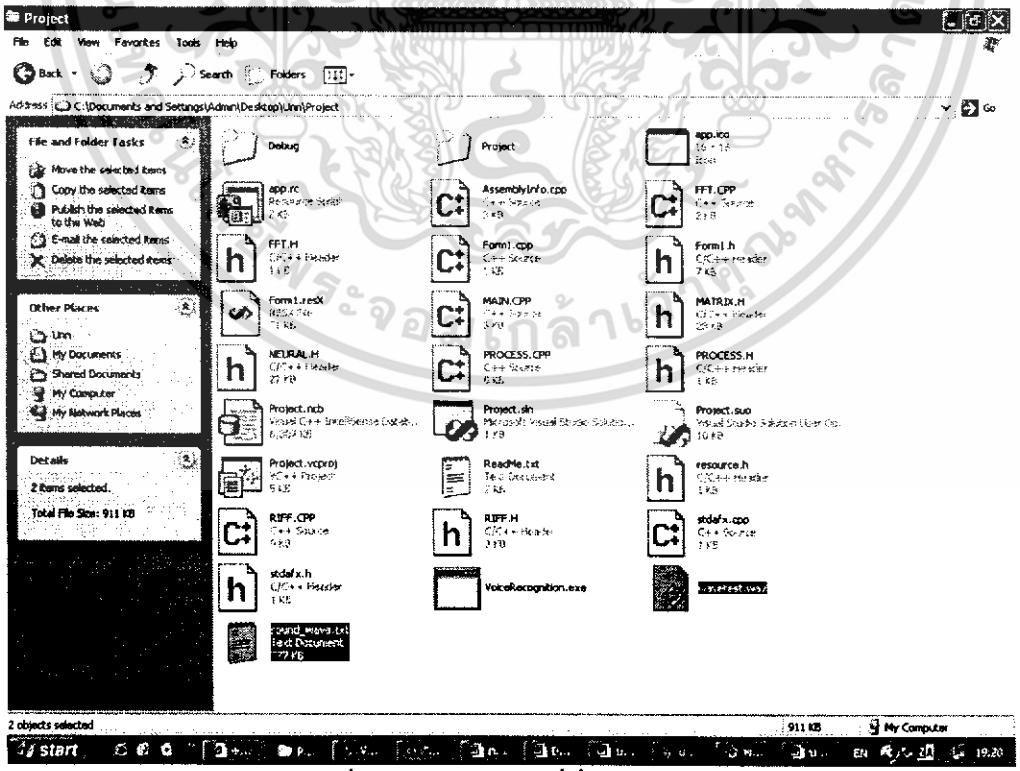
ระบบการสร้างรูปแบบอ้างอิง

1. เสียงทุกเสียงที่ต้องการสร้างเป็นรูปแบบอ้างอิงจะถูกรับผ่านทางไมโครโฟน จากผู้พูด 20 คน พูดคนละ 10 คำสั่ง ผ่านโปรแกรม Sound Recorder ที่มีมาพร้อมกับ Windows XP แล้ว

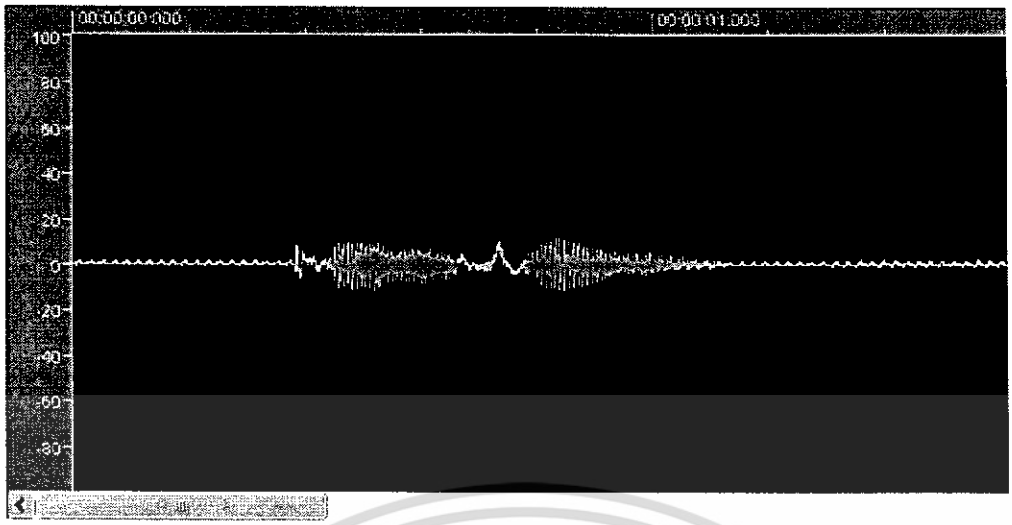


รูปที่ 1 แสดงการใช้งานโปรแกรม Sound Recorder

การบันทึกเสียงพูดเป็นข้อมูลดิจิตอลโดยใช้การ์ดเสียง จะทำให้ได้เวฟไฟล์ (.wav) ที่มีรูปแบบของไฟล์ RIFF (Resource International File Format) ซึ่งเป็นรูปแบบมาตรฐานของ MS windows ที่ใช้กันอย่างกว้างขวาง ซึ่งเวฟไฟล์ทุกเสียงจะถูกเก็บไว้ในไดเรกทอรี่เดียวกับโปรแกรมรู้จำ ชื่อ “wavetest.wav” และ “sound_wave.txt”

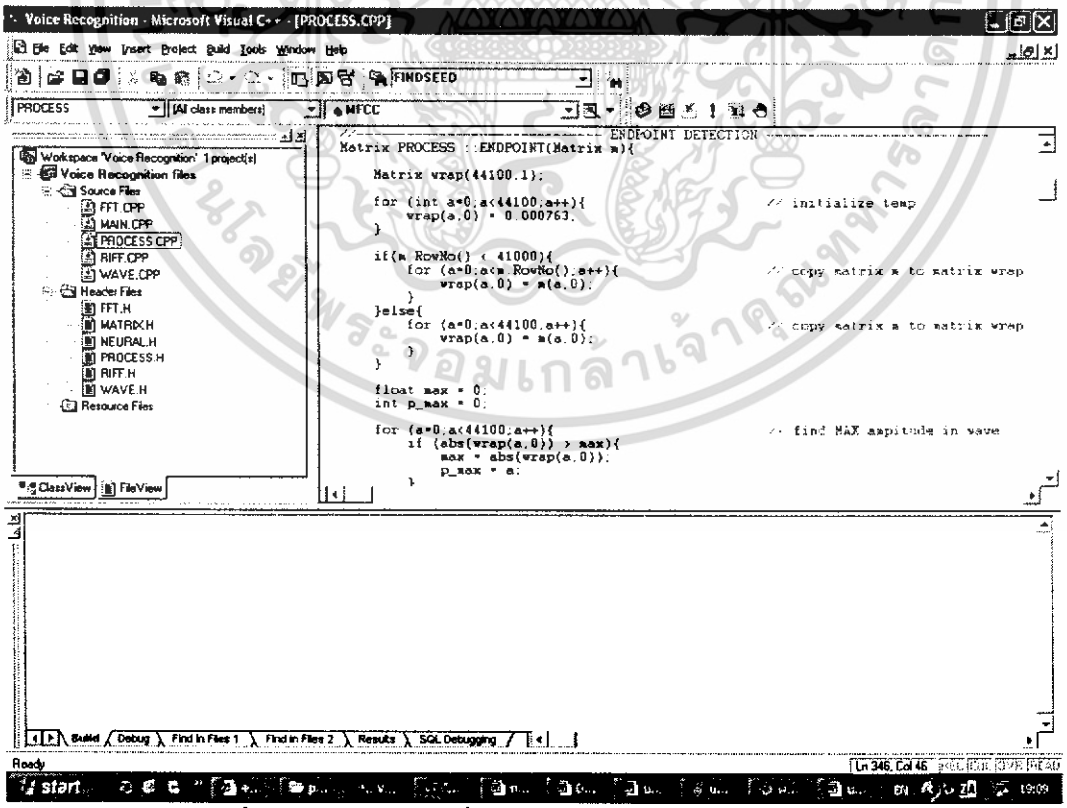


รูปที่ 2 แสดงไดเรกทอรี่ที่เก็บเวฟไฟล์



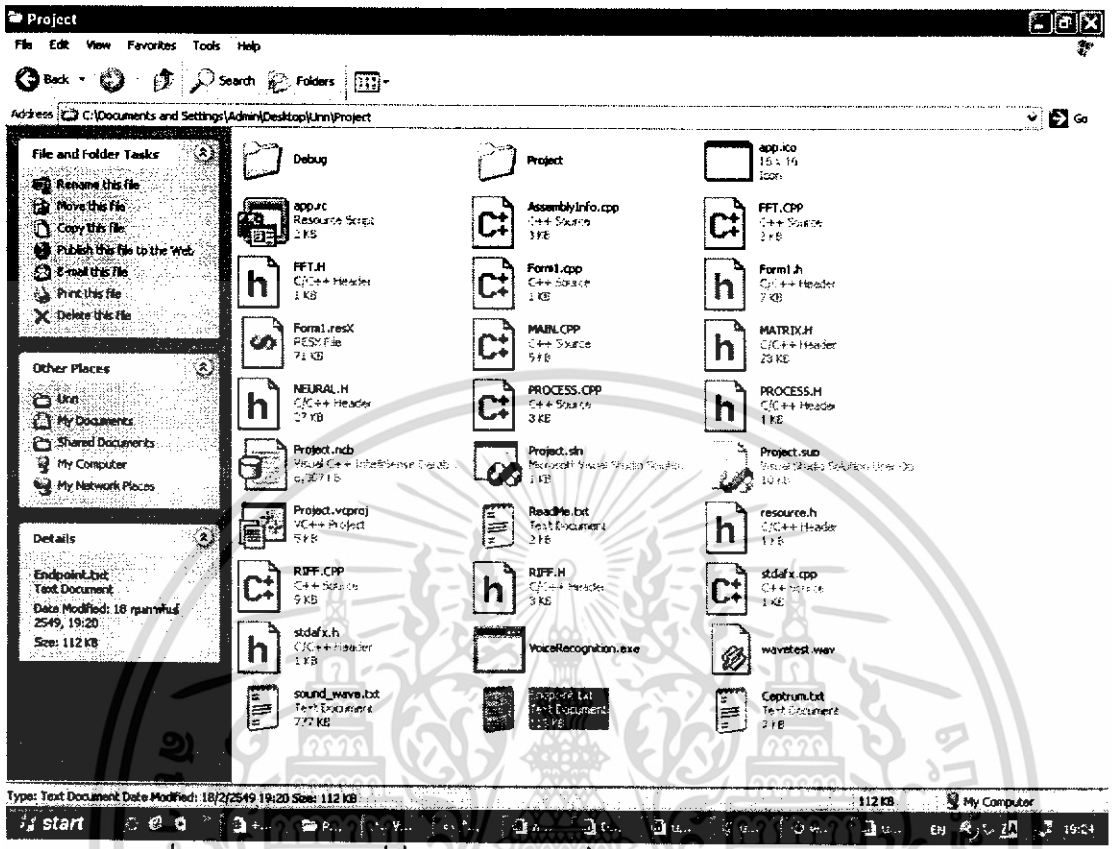
รูปที่ 3 แสดงรูปแบบเวฟไฟล์โดยแสดงค่าแอมปลิจูดเทียบกับเวลา

2. ข้อมูลเสียงทุกคำสั่งทำการกรองเสียงและตัดหัวท้ายเสียงโดยหาค่า amplitude ที่มากที่สุดของเสียงนั้นๆ และกำหนดจุดเริ่มต้นของเสียงโดยดูจากกำหนดค่าเริ่มต้นมีค่าเป็น 0.2 เท่าของค่าสูงสุด (โดยค่าสูงสุด 0.2 เท่า หาได้จากการทดลอง) หากจุดใดมีค่าเกินค่าเริ่มต้นก่อน จุดนั้นจะกลายเป็นจุดเริ่มต้น และจุดสุดท้ายจะห่างจากจุดเริ่มต้นเท่ากับ 10,000 แซมเปิ้ล ซึ่งทุกๆที่ถูกตัดหัวและท้ายเสียงจะมีความยาวของเสียงเท่ากันเท่ากัน กระบวนการทั้งหมดนี้เรียกว่า Pre – Processing ซึ่งในการสร้างรูปแบบอ้างอิงถูกนำมาใช้กับโปรแกรม Microsoft Visual Studio 6.0



รูปที่ 4 แสดงโปรแกรมที่ใช้ในกระบวนการ Pre - Processing

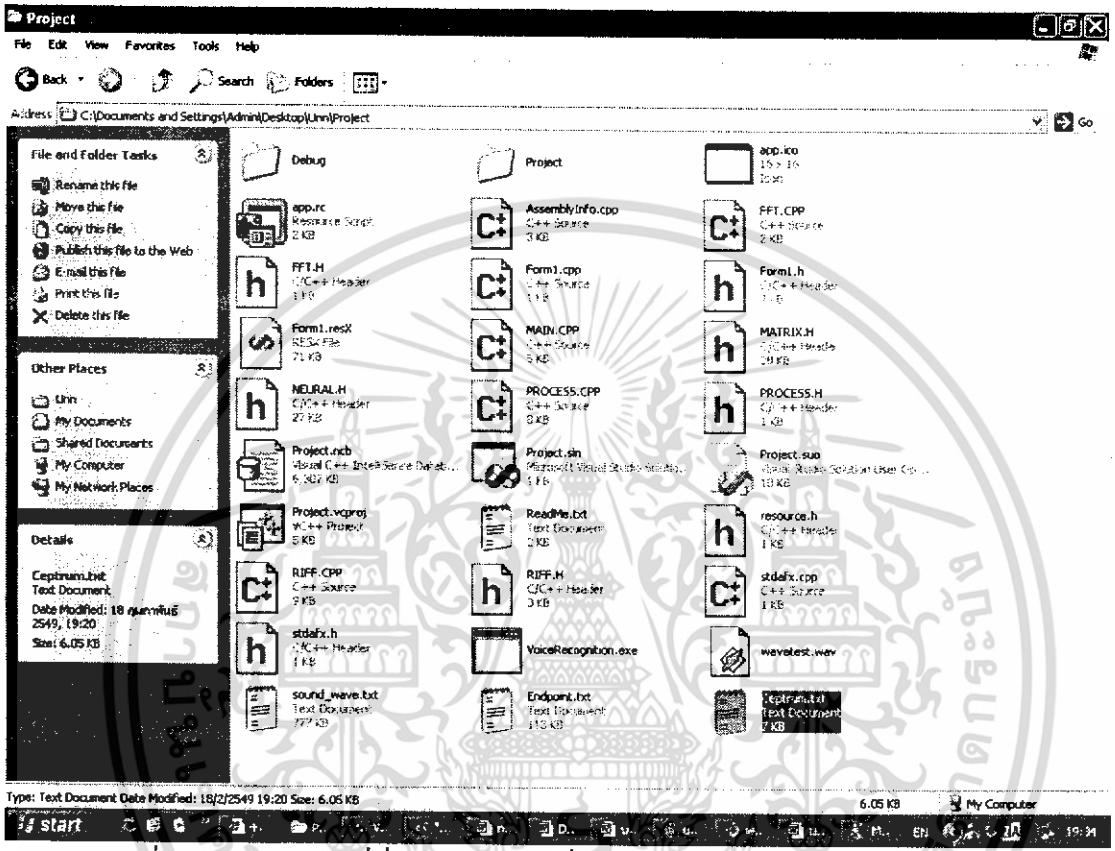
จากนั้น ข้อมูลของเสียงจะถูกเก็บไว้ที่ไดเรกทอรี่เดียวกันกับโปรแกรมการรู้จำเช่นกัน ชื่อ “Endpoint.txt”



รูปที่ 5 แสดงไดเรกทอรี่ที่เก็บข้อมูลเสียงที่ผ่านกระบวนการ Pre – Processing

รูปที่ 6 แสดงรูปแบบเวฟไฟล์ที่ผ่านกระบวนการ Pre – Processing

3. ข้อมูลเสียงของแต่ละคำสั่งที่ทำการกรองเสียงและตัดหัวท้ายเสียงเข้าสู่กระบวนการ Feature Extraction ซึ่งเป็นการสกัดค่าของเสียงโดยใช้วิธีเมลฟรีเควนซี่เซปตรัมโคเอฟฟีเชี่ยน (MFCC) โดยเสียงทุกเสียงจะถูกเก็บไว้ในรูปแบบไฟล์ (.txt) ชื่อ Ceptrum.txt ซึ่งอยู่ไดเรกทอรี่เดียวกันกับโปรแกรมการรู้จำ



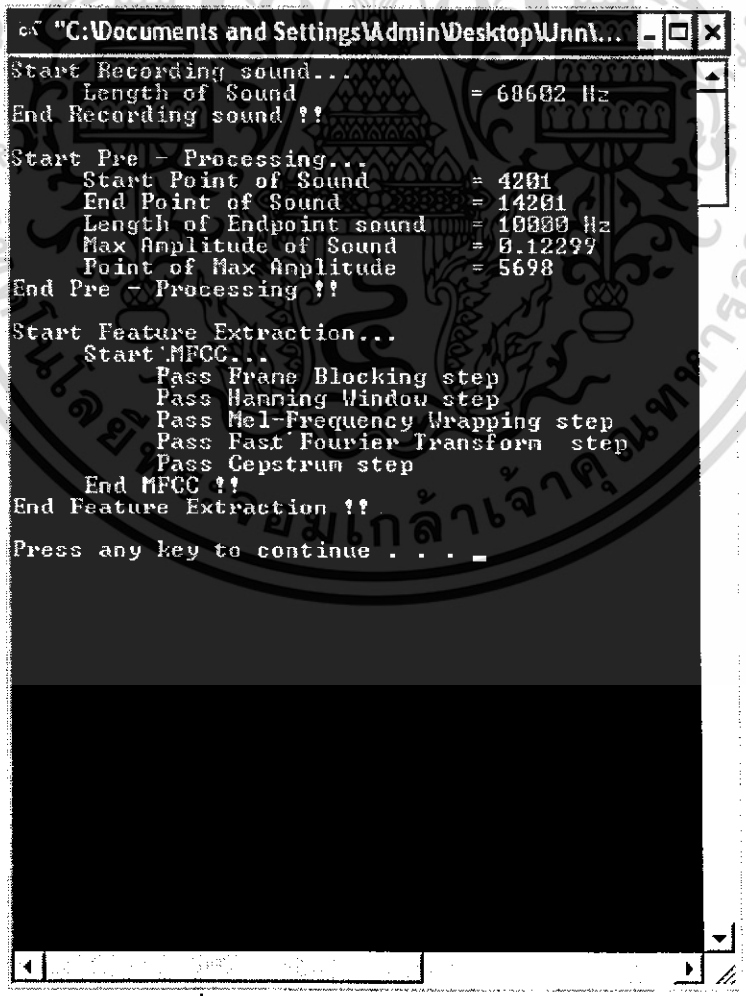
รูปที่ 7 แสดงไดเรกทอรี่ที่เก็บข้อมูลเสียงที่ผ่านกระบวนการ Feature Extraction

หลังจากสกัดค่าของเสียง แต่ละเสียงจะมีพารามิเตอร์ทั้งหมด 646 พารามิเตอร์ (38 x 17) ซึ่งถือว่าเป็นค่า feature ของเสียงแต่ละเสียง

-1.543199	-2.778970	2.217417	5.046230	3.476149	3.638813	-1.878960	2.040574	-1.229735	-5.516
-1.972613	-5.405566	-5.981709	-4.232970	-6.107317	-1.768039	2.360300	0.453747	-6.419863	-4.410
-0.794962	-6.773705	-8.905374	-9.111171	-8.248025	-5.028241	-0.492517	0.211940	-3.344425	-3.570
7.829164	6.014104	5.378962	3.051023	6.155446	5.682274	2.181398	4.405239	8.191285	8.4054
0.056615	-1.750892	-2.914176	-2.938580	-2.814151	-1.818418	0.797869	0.019190	-1.166942	-0.646
0.332368	0.520784	1.972333	3.715848	2.527862	2.761374	1.727676	0.991512	0.172156	-0.384
-0.476221	-1.724768	-1.279432	-2.555524	-2.882242	-2.301145	0.778775	-1.589753	-2.582057	-1.303
1.307558	1.126414	-0.452347	-1.169575	0.313224	0.485521	1.318399	0.485843	0.944543	0.9022
-0.161493	-0.454124	1.058511	1.189332	2.574611	2.098523	0.669376	1.168459	-0.284823	-2.532
1.317775	-0.723276	-0.309608	1.110186	0.984149	1.697843	1.204179	1.186373	-0.423974	-0.674
-1.272566	-1.803458	-1.845704	0.179100	-1.578659	-1.304882	0.199256	-0.467337	-1.423990	-0.404
0.333630	0.193126	-0.886826	0.068662	-0.617381	0.016989	1.041595	0.981228	0.421399	-0.177
1.061348	0.669739	-0.924275	-1.297021	-0.380258	-0.752609	1.135722	0.760252	0.515875	1.4605
-1.519758	-1.856269	-0.781948	-2.166442	-0.545216	0.726215	1.350512	0.472805	-1.533586	-1.957
0.738115	1.291827	1.725108	0.390660	1.077424	2.048709	-0.381597	1.513946	1.463230	0.9352
-0.641499	-1.641738	-0.610141	-0.883249	-2.695029	-1.016121	0.134013	-0.619602	-1.678512	-0.364
-0.256766	0.011695	0.649521	0.688309	-1.002921	0.043926	-0.216616	2.017273	0.324821	0.6317
-0.007470	-0.287850	-0.742602	-1.044923	-1.852434	-0.652260	0.912454	0.543445	0.335944	0.8087
-0.797802	-1.136535	-1.416240	-1.104479	-0.603590	-0.224847	0.707765	0.485307	-0.835382	-0.629
0.669069	0.176569	0.050049	0.499816	0.414373	1.248993	-0.262694	0.832684	-0.145952	-0.314
-0.430708	-0.982807	-1.115578	-0.624868	-0.764807	-0.375257	0.205004	-0.174795	-1.002343	-0.961
0.015744	0.263946	0.575068	0.682822	0.348563	0.263749	1.042727	1.436894	0.742920	-0.333
-0.449145	-0.063955	0.250660	-0.037461	0.640032	-0.398434	-0.100725	-0.491219	0.248746	-0.672
1.021028	0.643266	0.152490	-0.513472	0.213466	-0.442560	-0.056802	0.014216	0.261794	-0.227
-0.186266	0.449238	0.241725	0.646467	1.663203	1.112013	0.937444	0.683852	0.431093	-0.705
0.773352	1.027104	0.128223	0.528812	2.057156	1.378301	0.085294	0.216301	0.880576	0.0653
0.481177	0.474004	0.134967	1.513235	2.174572	0.927589	-0.383336	0.073964	0.655339	0.5081
0.407370	1.310616	0.942983	1.685442	2.080450	0.348011	-0.647070	-0.027515	1.288905	0.6881
0.830932	1.120313	1.429629	1.098183	0.800054	-0.517364	-0.207093	0.361682	1.132051	0.9476
-0.126547	0.797176	1.637899	1.045370	-0.020177	-0.191097	0.476868	0.490873	0.723988	0.9186
-0.112655	0.860625	1.250331	0.550769	-1.076942	-0.686112	0.138502	0.175630	0.482561	1.5571

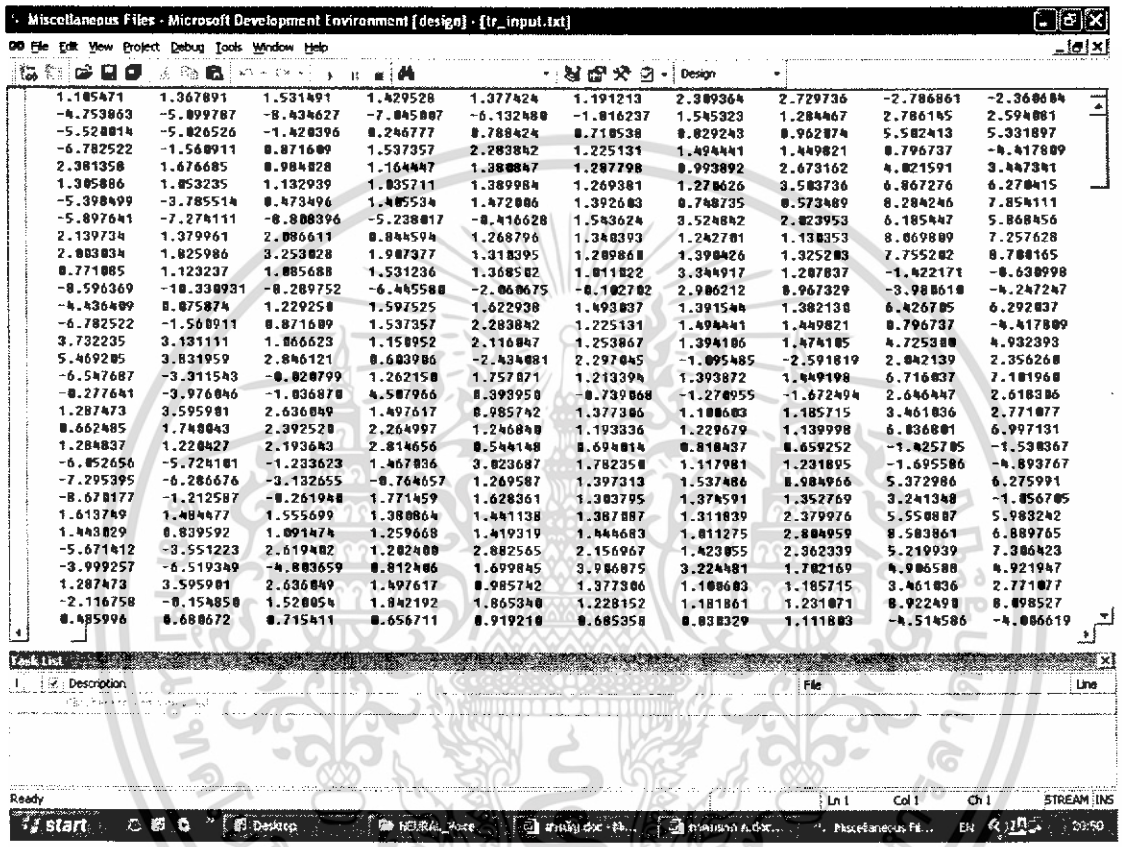
รูปที่ 8 แสดงค่าพารามิเตอร์ของเสียงแต่ละเสียง ขนาด 38x17

โปรแกรมจะแสดงผลการสกัดค่าทั้งหมดตั้งแต่ขั้นที่ (1) ถึงขั้นที่ (3) เช่นค่าความยาวของเสียงตั้งต้น ค่าแอมปลิจูดสูงสุด กระบวนการสกัดค่าในแต่ละขั้นตอน เป็นต้น



รูปที่ 9 ผลของการสกัดค่าลักษณะสำคัญ

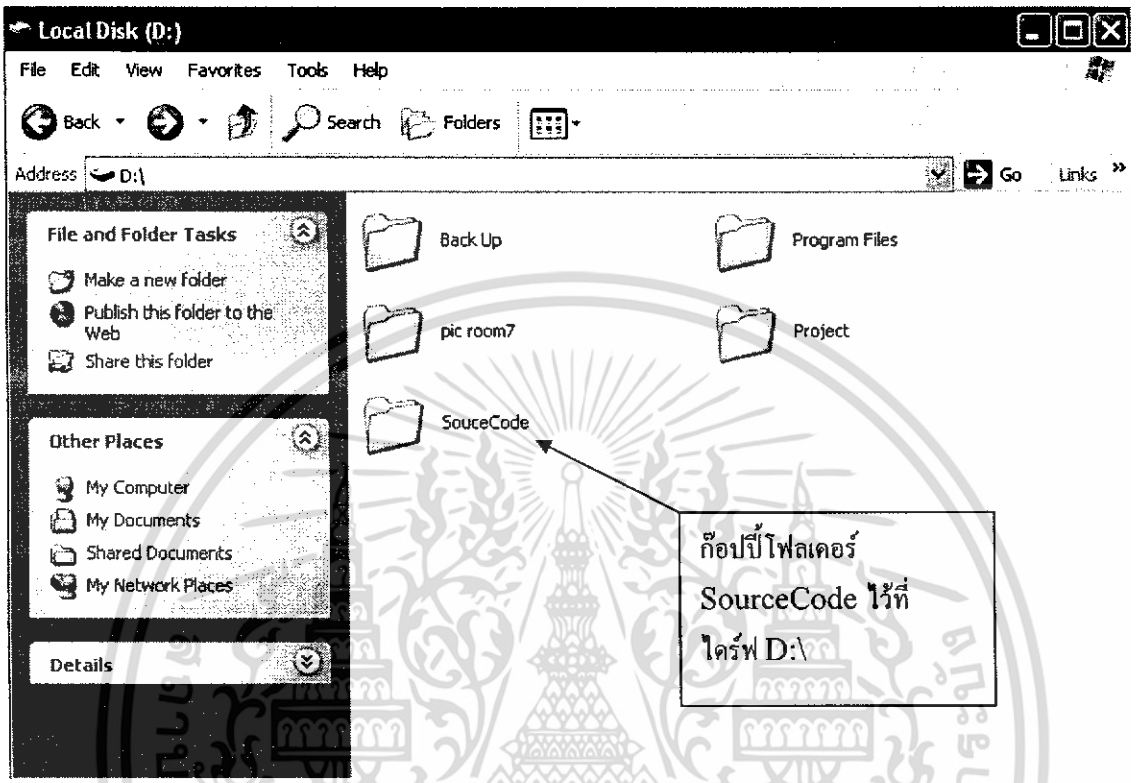
4. ทำการรวมค่าพารามิเตอร์ของแต่ละเสียงโดย 1 เสียงมีขนาดเท่ากับ 1 แถว 646 หลัก ตามจำนวนพารามิเตอร์ของเสียงทุกเสียง ดังนั้น ถ้าต้องการสร้างรูปแบบอ้างอิงของเสียง n เสียง ขนาดของรูปแบบอ้างอิงจะมีค่าเท่ากับ n x 646 หากรูปแบบอ้างอิงมีจำนวนเสียงมาก การรู้จำก็จะมีประสิทธิภาพดีขึ้นเช่นกัน แต่ข้อเสียคือ จะใช้เวลานานมากยิ่งขึ้นในการปรับปรุงค่าน้ำหนักในการเรียนรู้แบบ Back – Propagation



ภาคผนวก ข.
การติดตั้งและการใช้งานโปรแกรม

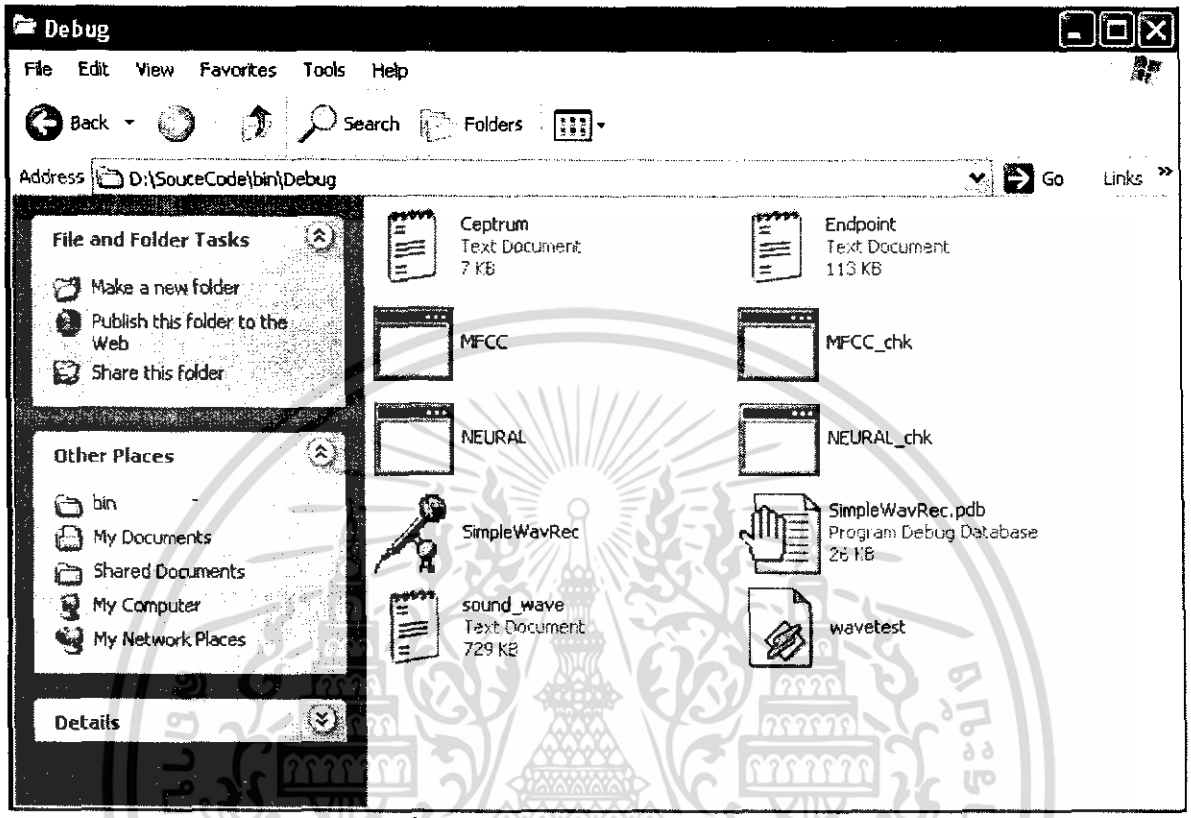
การติดตั้งโปรแกรม

1. นำไฟล์ SourceCode มาวาไว้ที่ไดเรกทอรี่ D:\



รูปที่ 1 แสดงการติดตั้งโปรแกรม

2. เข้าไปที่ไดเรกทอรี่ D:\SourceCode\bin\Debug จะพบโปรแกรมรู้จำเสียง โดยสามารถใช้โปรแกรมได้ที่นี่



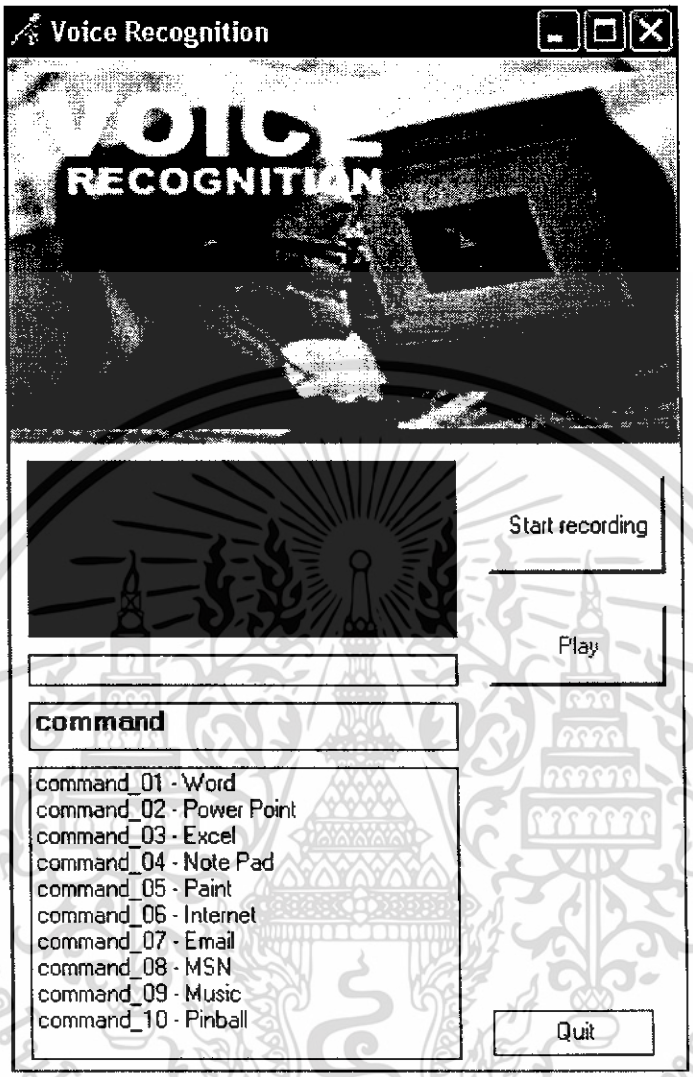
รูปที่ 2 แสดงไฟล์ต่างๆ ในโปรแกรม

File Name	Description
MFCC.exe	สกัดค่าลักษณะสำคัญ
MFCC_chk.exe	สกัดค่าลักษณะสำคัญแบบแสดงรายละเอียดขั้นตอนต่างๆ
NEURAL.exe	โหลดข้อมูลและจำแนกเสียง

NEURAL_chk.exe	โหลดข้อมูลและจำแนกเสียงแบบแสดงรายละเอียดขั้นตอนต่างๆ
wavetest.wav	ไฟล์เสียง .wav ที่บันทึกไว้
sound_wave.txt	แสดงค่าสเปคตรัมของเสียงทั้งหมด
Endpoint.txt	แสดงค่าสเปคตรัมของเสียงหลังจากผ่านการตัดหัวท้ายเสียง
Ceptrum.txt	แสดงค่า Ceptrum ที่ได้จากการสกัดโดย MFCC
SimpleWavRec.exe	โปรแกรมรู้จำแนกเสียง

ตารางที่ 1 แสดงชื่อไฟล์และหน้าที่ของแต่ละไฟล์

การใช้โปรแกรม



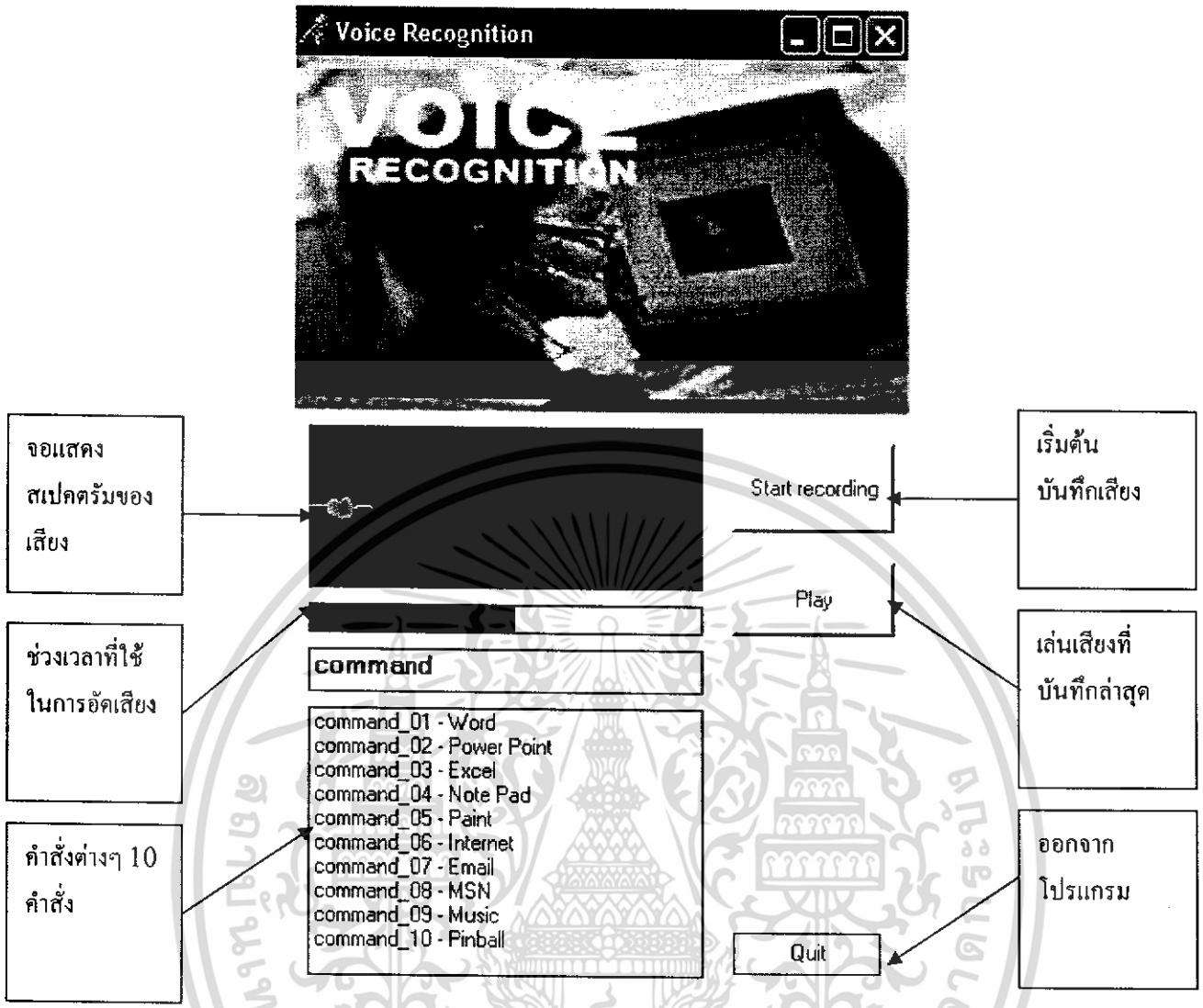
รูปที่ 3 แสดงรูปแบบของโปรแกรม voice recognition

ขั้นตอนการใช้งานโปรแกรม

1. เปิดโปรแกรม SimpleWavRec ใน D:\SouceCode\bin\Debug\
2. กดปุ่ม Start recording เพื่ออัดเสียงที่จะนำไปใช้ในการวิเคราะห์
3. กดปุ่ม Play เพื่อทดสอบเสียงที่เพิ่งจะบันทึกล่าสุด
4. ผลลัพท์ที่ได้จากจำแนกเสียง จะนำไปเปิดโปรแกรมนั้นออกมา

Commands	Results
1. Word	เปิดโปรแกรม Microsoft Word
2. Power Point	เปิดโปรแกรม Microsoft Power Point
3. Excel	เปิดโปรแกรม Microsoft Excel
4. Note Pad	เปิดโปรแกรม Note Pad
5. Paint	เปิดโปรแกรม Paint
6. Internet	เปิดโปรแกรม Internet Explorer
7. Email	เปิดโปรแกรม Microsoft Outlook Express
8. MSN	เปิดโปรแกรม MSN Messenger
9. Music	เปิดโปรแกรม Windows Media Player
10. Pinball	เปิดโปรแกรมเกมส์ Pinball

ตารางที่ 2 แสดงคำสั่งต่างๆ และผลลัพท์ที่ได้จากการรู้จำ



รูปที่ 4 แสดงฟังก์ชั่นต่างๆของโปรแกรม

Buttons	Functions
Start recording	อัดเสียงเป็นไฟล์ชื่อ wavetest.wav โดยมี Format เป็น 11025 Hz,Mono,8 Bits ใน โฟลเดอร์เดียวกับโปรแกรม voice recognition
Play	เล่นไฟล์ wavetest.wav ที่บันทึกล่าสุด
Quit	ออกจากการใช้โปรแกรม

ตารางที่ 3 แสดงปุ่มต่างๆ และหน้าที่ของแต่ละปุ่ม

ภาคผนวก ค

ซอร์สโค้ดคลาสการทำงานของ Back – Propagation neural network

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <math.h>                    //For tanh

// returns a float in the range -1.0f -> - 1.0f
#define RANDOM_CLAMP    (((float)rand()-(float)rand())/RAND_MAX)

//returns a float between 0 & 1
#define RANDOM_NUM      ((float)rand()/(RAND_MAX+1))

using namespace std;

//*************************************************************************
class Dendrite {
      public:
            double d_weight;                  //Weight of the neuron
            unsigned long d_points_to;

            Dendrite(double weight=0.0, unsigned long points_to=0){
            //Constructor
                  d_weight    = weight;
                  d_points_to = points_to;
            //Give it a initial value
            }
};
//*************************************************************************
class Neuron  {
      public:
            unsigned long n_ID;//ID of a particular neuron in a layer
            double n_value;          //Value which Neuron is holding
            double n_bias;           //Bias of the neuron
            double n_delta;          //Used in back  prop
            Dendrite *Dendrites;     //Dendrites
      //Constructor assigning initial values
      Neuron(unsigned long ID=0,double value=0.0,double bias=0.0){
            n_ID    = ID;
            n_value = value;
            n_bias  = bias;
            n_delta = 0.0;
      }

      void SetDendrites(unsigned long dendrite){
                  Dendrites=new Dendrite[dendrite];

            for(int i=0;i<dendrite;i++){  // Initialize the dendrite
                  Dendrites[i].d_points_to=i;
            }
      }
};
//*************************************************************************
class Layer{
      public:
            Neuron *Neurons;         //Pointer to array of neurons

            void Initialize(unsigned long size) {
                  Neurons = new Neuron [size];
            }
            ~Layer(){
                  delete Neurons;
            }
```

```
            Neuron GetNeuron(unsigned long index){//Give the neuron
                  return Neurons[index];
            }

            void SetNeuron(Neuron neuron,unsigned long index){
                  Neurons[index]=neuron;
            }
};
//*************************************************************
class Network {          //***** The real neural network *****
   public:
      double net_learning_rate;           //Learning rate of network
      Layer *Layers;                      //The layers in network
      unsigned long net_tot_layers;       //Number of layers
      double *net_inputs;                 //Input array
      double *net_outputs;                //Output layers
      unsigned long *net_layers;          //Array which tells no. of
                                          neurons in each layer
      Network() {}
      //*************************************************************
      //---------------------------- Load Weight  ---------------------
      void LoadWBij (void) {
         ........
      }
      //-------------------------------------------------------------
      void LoadWBjk (void) {
         ........
      }
      //-------------------------------------------------------------
      void LoadBiasJ (void) {
         ........
      }
      //-------------------------------------------------------------
      void LoadBiasK (void) {
         ........
      }
      //*************************************************************v
      void InitWB(void){              //Randomize weights and biases
         int i,j,k;

         for(i=0;i<net_tot_layers;i++){
            for(j=0;j<net_layers[i];j++){  //Last layer does not
               if(i!=(net_tot_layers-1)){  //require weights
                  Layers[i].Neurons[j].SetDendrites(net_layers[i+1]);

                  for(k=0;k<net_layers[i+1];k++){
                 Layers[i].Neurons[j].Dendrites[k].d_weight=GetRand();
                  }
               }
               if(i!=0){                  //First layer does not need biases
                  Layers[i].Neurons[j].n_bias=GetRand();
               }
            }
         }
      }
      //*************************************************************
```

```
//------------------------------ Record file --------------------
void ReccordWBij(void){
   ........
}
//*********************************************************************
void ReccordWBjk(void){
   ........
}
//*********************************************************************
void ReccordBiasJ(void){
   ........
}
//*********************************************************************
void ReccordBiasK(void){
   ........
}
//*********************************************************************
int SetData(double learning_rate,unsigned long layers[],unsigned
            long tot_layers) {//Function to set various
                              //parameters of the net
   if (tot_layers<2) return(-1);//If total no. of layers < 2

   net_learning_rate=learning_rate;
   net_layers= new unsigned long [tot_layers];
   Layers=new Layer[tot_layers];

   for(int i=0;i<tot_layers;i++){
      net_layers[i]=layers[i];
      Layers[i].Initialize(layers[i]);
   }

   net_inputs=new double[layers[0]];
   net_outputs=new double[layers[tot_layers-1]];
   net_tot_layers=tot_layers;

   return 0;
}
//---------------------- Load Input to neurons ------------------
int SetInputs(double inputs[]){     //Function to set the inputs
   for(int i=0;i<net_layers[0];i++){
      Layers[0].Neurons[i].n_value=inputs[i];
   }
   return 0;
}
//---------------------- Return output to compare ---------------
double * GetOutput(void){      //Gives the output of the net
   double *outputs;
   int i,j,k;
   outputs=new double[net_layers[net_tot_layers-1]];

   for(i=1;i<net_tot_layers;i++){
      for(j=0;j<net_layers[i];j++){
         Layers[i].Neurons[j].n_value=0;
         for(k=0;k<net_layers[i-1];k++){
            Layers[i].Neurons[j].n_value=Layers[i].Neurons[j].
            n_value+Layers[i-1].Neurons[k].n_value
            *Layers[i-1].Neurons[k].Dendrites[j].d_weight;
         }
         Layers[i].Neurons[j].n_value=Layers[i].Neurons[j].
               n_value+Layers[i].Neurons[j].n_bias;     //Add bias
```

```
            Layers[i].Neurons[j].n_value=Limiter(Layers[i].
                    Neurons[j].n_value);    //Squash that value
         }
      }
      for(i=0;i<net_layers[net_tot_layers-1];i++){
         outputs[i]=Layers[net_tot_layers-1].Neurons[i].n_value;
      }
      return outputs;                       //Return the outputs
}

void Update(void){                          //Just a dummy function
      GetOutput();
}
//---------------------------------------------------------------------
double ChkStop (double target[]){
   double Target, Actual, Delta;
   Target=target[0];                                  //Target value
   Actual=Layers[2].Neurons[0].n_value;               //Actual value
   Delta= abs(Target - Actual);
   return Delta;
}
//---------------------------------------------------------------------
double Limiter(double value){       //Limiter no limit value
                        between 1 and -1 (sigmoid function)
   return (1.0/(1+exp(-value)));
}
//---------------------------------------------------------------------
double GetRand(void){
   time_t timer;
   struct tm *tblock;
   timer=time(NULL);
   tblock=localtime(&timer);

   int seed=int(tblock->tm_sec+100*RANDOM_CLAMP+100*RANDOM_NUM);

   srand(seed);
   return ((RANDOM_CLAMP+RANDOM_NUM)/286);
}
//---------------------------------------------------------------------
double SigmaWeightDelta(unsigned long layer_no,
                        unsigned long neuron_no){
   double result=0.0;
   for(int i=0;i<net_layers[layer_no+1];i++) {
      result=result+Layers[layer_no].
            Neurons[neuron_no].Dendrites[i].d_weight
            *Layers[layer_no+1].Neurons[i].n_delta;
   }
   return result;
}
//---------------------------------------------------------------------
int Train(double inputs[],double outputs[]){
//The standard Backprop Learning algorithm
   int i,j,k;
   double Target, Actual, Delta, Err;
   Err = 0;
   SetInputs(inputs);                   //Set the inputs
   Update();                            //Update all the values
```

```
            for(i=net_tot_layers-1;i>0;i
               for(j=0;j<net_layers[i];j++) {      //Go thru every neuron
               if(i==net_tot_layers-1){            //** in Output layer
                  Target=outputs[j];               //Target value
                  Actual=Layers[i].Neurons[j].n_value;   //Actual value
                  Delta= (Target - Actual) * Actual * (1 - Actual);
                  Layers[i].Neurons[j].n_delta=Delta;
                  for(k=0;k<net_layers[i-1];k++) {
                     Layers[i-1].Neurons[k].Dendrites[j].d_weight +=
                           Delta*net_learning_rate*\
                           Layers[i-1].Neurons[k].n_value;
                  }
                  Layers[i].Neurons[j].n_bias =
                     Layers[i].Neurons[j].n_bias
                     + Delta*net_learning_rate*1;

                  Err += ((Target - Actual)*(Target - Actual));

               }
               else {                                //In Hidden layer
                  Actual=Layers[i].Neurons[j].n_value;   //Actual value
                  Delta=  Actual * (1 - Actual)* SigmaWeightDelta(i,j);
                  for(k=0;k<net_layers[i-1];k++){
                     Layers[i-1].Neurons[k].Dendrites[j].d_weight +=
                           Delta*net_learning_rate*
                           Layers[i-1].Neurons[k].n_value;
                  }
                  if(i!=0)                 //Input layer does not have a bias
                     Layers[i].Neurons[j].n_bias =
                           Layers[i].Neurons[j].n_bias+
                           Delta*net_learning_rate*1;
               }
               }
            }
            return Err;
         }
//       ~Network(){ delete Layers; }
};

//*************************************************************************
*************************

void main (void){

      Network my;

      unsigned long inp=646;
      unsigned long hid=20;
      unsigned long outp=10;
      unsigned long layers[3];

        layers[0]=inp;
      layers[1]=hid;
      layers[2]=outp;

        int i=0,j=0,k=0,count=0;

        double max;
        int pnt;

        double ERR,SQQERR;
```

```
  //double input[]={1,0};
  double *outputs;

unsigned long iter=0;

  my.SetData(0.2,layers,3);

  my.InitWB();

double tr_inp[200][646];

  double buff[38][17];
  double test[1][646];

  double cmd[10][10]  =       {
{1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},

{0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},

{0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0},

{0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0},

{0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0},

{0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0},

{0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0},

{0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0},

{0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0},

{0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0}};

  char ch;

  //*************************************
  for(i=0 ; i<1 ; i++){
            char* aa;
            string axx,str;
            aa = "tr_input.txt";

            string constructorString (myread(aa));

            int x = 0;
            int y = 0;

            axx = myread("tr_input.txt");


            const char* str2;

            j=0;
            k=0;

            for(int s = 0; s < filesize("tr_input.txt"); s++){

                  if ( constructorString[s] != '\t' &&
constructorString[s] != '\n' ){x++;}
```

```
                        else {
                              if (constructorString[s] == '\n')

                                    x = x-1;

                              str   = axx.substr(y,x);
                              str2= str.c_str();
            // const char

                              float       tempMatrix = atof(str2);


                                    tr_inp[j][k]      = tempMatrix;


                              y = y+x+1;
                              x = 0;

                              k++;

                              if(k==646)

                              {
                                    j++;
                                    k=0;
                              };

                              if (constructorString[s] == '\n')

                                    y = y+1;

                        }
                  }
            }
            //************************************
            //************************************
            for(i=0 ; i<1 ; i++){
                  char* aa;
                  string axx,str;
                  aa = "Ceptrum0.txt";

                  string constructorString (myread(aa));

                  int x = 0;
                  int y = 0;

                  axx = myread("Ceptrum0.txt");


                  const char* str2;

                  int j=0,k=0;

                  for(int s = 0; s < filesize("Ceptrum0.txt"); s++){

                        if ( constructorString[s] != '\t' &&
constructorString[s] != '\n' ){x++;}
                        else {
                              if (constructorString[s] == '\n')

                                    x = x+1;
```

```
                                str    = axx.substr(y,x);
                                str2= str.c_str();
                // const char

                                float       tempMatrix = atof(str2);


                                buff[j][k]  = tempMatrix;


                                y = y+x+1;
                                x = 0;

                                k++;

                                if(k==17)

                                {
                                        j++;
                                        k=0;
                                };

                                if (constructorString[s] == '\n')

                                        y = y+1;

                        }
                    }

                    for(i=0;i < 38;i ++){
                        for(j=0;j<17;j++){
                            test[0][k] = buff[i][j];
                            k++;
                        }
                    }
            }
            //*****************************************


        cout << "<L>    for Loading database \n";
        cout << "<R>    for Recognition \n";
        cout << "<T>    for Training \n";
        cout << "<Q>    for Quit program \n";                    cout <<
"Press only 1 choice then Enter : ";
        cin >> ch;

        while (ch != 'q' && ch != 'Q'){

        //---------------------------- LOAD WEIGHT --------------------
----------

                if( ch == 'l' || ch == 'L'){
                    cout << "\nLoading database... \n";

                      my.LoadWBij();
                  my.LoadWBjk();
                      my.LoadBiasJ();
                      my.LoadBiasK();

                      cout << "Ending Loading";
```

```
                cout << "\n\n";
            }

        //------------------------------ TRAINING ----------------------
-----------

            if( ch == 't' || ch == 'T'){

                cout <<"Enter number of training Iterations : ";

                cin >>iter;

                cout<<"\nStarting Training... ";

                for(i=0;i<iter;i++){

                    k = 0;
                    count = 0;

                    //cout<<"\nTraining : "<<i+1;

                    for(j=0;j<200;j++){

                        ERR = my.Train(tr_inp[j],cmd[count]);

                        count++;

                        if(count > 9 ){
                            // k ++;
                            count = 0;
                        }

                        SQQERR += ERR;

                    }

                    k++;

                    SQQERR = SQQERR/2;

                    cout << count << " -> SQQERR = " << SQQERR
<<"\n";

                }

                //************************
                my.ReccordWBij();
                my.ReccordWBjk();
                my.ReccordBiasJ();
                my.ReccordBiasK();
                //************************

                cout<<"\nEnding Training. ";
            }

        //------------------------------ RECORD ------------------------
-----------

            if( ch == 'r' || ch == 'R'){
```

```
                    cout<<"\n\nStarting Testing... \n";

                      for(j=0;j<1;j++){

                            my.SetInputs(test[j]);

                            outputs=my.GetOutput();

                            for(i=0;i<outp;i++){

                                  cout<<"\nOutput"<<i<<" :
"<<outputs[i];

                            }

                            for(i=0;i<outp;i++){
                                  if (outputs[i] > max){
                                        max = outputs[i];
                                        pnt = i;
                                  }
                            }

                            if (pnt == 0) cout <<"\n\n You say : word ";
                            if (pnt == 1) cout <<"\n\n You say :
Powerpoint ";
                            if (pnt == 2) cout <<"\n\n You say : Excel
";
                            if (pnt == 3) cout <<"\n\n You say : NotePad
";
                            if (pnt == 4) cout <<"\n\n You say : Paint
";
                            if (pnt == 5) cout <<"\n\n You say :
Internet ";
                            if (pnt == 6) cout <<"\n\n You say : E-mail
";
                            if (pnt == 7) cout <<"\n\n You say : MSN ";
                            if (pnt == 8) cout <<"\n\n You say : Music
";
                            if (pnt == 9) cout <<"\n\n You say : Pinball
";

                            max = 0.0;

                            delete outputs;

                            double *outputs;
                      }

                      cout<<"\n\nEnd Testing.\n\n";
                }

                cout << "\n-----------------------------------------------
--------------------------------\n\n";
                cout << "<L>    for Loading database \n";
              cout << "<R>      for Recognition \n";
              cout << "<T>      for Training \n";
              cout << "<Q>      for Quit program \n\n";              cout
<< "Press only 1 choice then Enter : ";
              cin >> ch;
                cout << "\n";
```

```
        }
        cin.get();
}

//*****************************************************************************
*****************************

//********************************* Support File
*********************************

char * myread(char *filename)
{
      FILE *fp;



      fp = fopen(filename, "rb");
      fseek(fp, 0L, SEEK_END );
      size_t endPos = ftell( fp );

      fseek(fp, 0, SEEK_SET);

      char *tag_buffer;
      tag_buffer = (char*) malloc (endPos);

      fread(tag_buffer, sizeof(char),endPos, fp);
      fclose(fp);
      return tag_buffer;
}


int filesize(char *filename)
{
      FILE *fp;



      fp = fopen(filename, "rb");
      fseek(fp, 0L, SEEK_END );
      size_t endPos = ftell( fp );
      fclose(fp);
      return endPos;
}

//*****************************************************************************
*********
```

ภาคผนวก ง

ซอร์สโค้ดคลาสการทำงานของ Pre – processing และ Feature Extraction

```
class PROCESS
{
      private:
            Matrix MELFILTERBANK(double p,double n,double fs);
            Matrix HAMMING(int n);
            Matrix DCT(Matrix x,double n);
            Matrix FFT_DCT(Matrix b,Matrix m,int order);
            Matrix MFCC(Matrix s,double fs);
      public:
            Matrix EXTRACT(Matrix wavtrain,int length_wav,int
            no_cmd);
            Matrix ENDPOINT(Matrix m);
};
//*****************************************************************************
```

Pre – Processing

```
//*****************************************************************************
Matrix PROCESS ::ENDPOINT(Matrix m){

      Matrix wrap(44100,1);

      for (int a=0;a<44100;a++){          //initialize temp
            wrap(a,0) = 0.000763;
      }

      if(m.RowNo() < 41000){
            for (a=0;a<m.RowNo();a++){    //matrix m to matrix wrap
                  wrap(a,0) = m(a,0);
            }
      }else{
            for (a=0;a<44100;a++){
            wrap(a,0) = m(a,0);
            }
      }

      float max = 0;
      int p_max = 0;

      for (a=0;a<44100;a++){          // find MAX amplitude in wave
            if (abs(wrap(a,0)) > max){
                  max = abs(wrap(a,0));
                  p_max = a;
            }
      }

      float cut_point = 0.2*max;     // define threshold valur

      a = 0;

      while (abs(wrap(a,0)) < cut_point){
            a++;
      }

      int st;
      st = a;                         // start point of wave

      Matrix temp(10000,1);           // creat matrix has length 10000
      int x = 0;
      int y = 0;
```

```
        for (int a2 = st ; a2 < st+10000 ; a2++){
              temp(x,y) = wrap(a2,0);
              x++;
        }
        return temp;
}
//*****************************************************************************
```

Feature Extraction

```
//*****************************************************************************
//------------------------------- MFCC -------------------------------
Matrix PROCESS ::MFCC(Matrix s,double fs) {
        int N = 1024;       // N = pow(2,order)
        int M = N/2;
        int nof = 40;
        int length,cols,row,col;
        double fin;

        length = s.RowNo();
        fin = length-N+1;
        cols = floor(fin/(N-M));

        Matrix a;
        a.Null(N,cols);

        for (row=0; row<N; row++) {
              a(row,0) = s(row,0);
        }

        for (col=1; col<cols; col++) {
              for (row=0; row<N; row++) {
                    a(row,col) = s((N-M)*col+row,0);
              }
        }
        //------ Hamming Window -------
        Matrix h(N,1),b(N,cols);
        h = HAMMING(N);

        for (col=0; col<cols; col++) {
              for (row=0; row<N; row++) {
                    b(row,col) = a(row,col)*h(row,0);
              }
        }
        //------ Melfilterbank ------
        Matrix m;
        m = MELFILTERBANK(nof,N,fs);

        Matrix v;
        int order = 0;

        while ( N != pow(2,order) ) {
              order++;
        }

        v = FFT_DCT(b,m,order);

        return v;
}
```

```
//--------------------------- Hamming Window -------------------------
Matrix PROCESS ::HAMMING(int n){
      double arg;
      int t;
      Matrix h(n,1);

      arg = 2.0*PI/(n-1);
      t = n/2;

      for (int j = 0; j < n; j++) {
            h(j,0) = 0.54 - 0.46*cos(arg*j);
      }
      return h;
}
//--------------------------- Melfilterbank -------------------------
Matrix PROCESS ::MELFILTERBANK(double p,double n,double fs){
      double f0 , fn2 , lr;
      double b1,b2,b3,b4;
      int i;
      Matrix a(4,1);

      f0 = 700/fs;
      fn2 = floor(n/2);
      lr = log(1 + 0.5/f0) / (p+1);

      a(0,0) = 0;
      a(1,0) = 1;
      a(2,0) = p;
      a(3,0) = p+1;
      for (i = 0; i<4; i++) {
            a(i,0) = n*(f0*(exp(a(i,0)*lr)-1));
      }
      b1 = floor(a(0,0)) + 1;
      b2 = ceil(a(1,0));
      b3 = floor(a(2,0));
      b4 = __min(fn2,ceil(a(3,0)))-1;

      Matrix pf(1,b4),fp(1,b4),pm(1,b4);
      for (i=b1; i<=b4; i++) {
            pf(0,i-1) = log(1 + (i/n/f0)) / lr;
            fp(0,i-1) = floor(pf(0,i-1));
            pm(0,i-1) = pf(0,i-1) - fp(0,i-1);
      }

      int cols = b3+b4-b2+b1;
      int temp=0;
      int row,col;
      Matrix r(1,cols),c(1,cols),v(1,cols),m(p,1+fn2);

      for (col=b2; col<=b4; col++) {
            r(0,col - b2) = fp(0,col-1);
      }

      for (col=1; col<=b3; col++) {
            temp = b4+col-b2;
            r(0,temp) = 1 + fp(0,col-1);
      }

      for (col=b2; col<=b4; col++) {
            c(0,col-b2) = col + 1;
      }
```

```
        for (col=1; col<=b3; col++) {
              temp = b4 - b2 + col ;
              c(0,temp) = col + 1;
        }

        for (col=b2; col<=b4; col++) {
              v(0,col-b2) = 2*(1 - pm(0,col-1));
        }

        for (col=1; col<=b3; col++) {
              temp = b4 - b2 + col;
              v(0,temp) = 2*(pm(0,col-1));
        }

        m.Null(p,1+fn2);
        for (i=0; i<b3+b4-b2+b1; i++) {
              row = r(0,i)-1;
              col = c(0,i)-1;
              m(row,col) = v(0,i);
        }

        return m;
}
// ------------------------------ FFT -------------------------------
Matrix PROCESS ::FFT_DCT(Matrix b,Matrix m,int order) {
        int rows = b.RowNo();
        int N = rows;
        int cols = b.ColNo();
        int r=0,c=0;
        int n2;

        n2 = 1+floor(N/2);

        Matrix y(n2,1);
        Matrix abs_y(n2,1);
        Matrix ms(m.RowNo(),1);
        Matrix log_ms(ms.RowNo(),1);
        Matrix dct_o(log_ms.RowNo(),1);
        Matrix v(dct_o.RowNo()-2,cols);

        double temp = 0;
        double* re = new double[rows];
        double* im = new double[rows];

        for(c=0; c<cols; c++) {
              for (r=0; r<rows; r++){
                    re[r] = b(r,c);
                    im[r] = 0.0;
              }

              fft(re,im,order,FFT);

              for (r=0; r<n2; r++) {
                    y(r,0) = sqrt(pow(re[r],2) + pow(im[r],2));
              }

              abs_y = y ^ 2;

              ms = m * abs_y;
```

```
            for (r=0; r<ms.RowNo(); r++) {
                  log_ms(r,0) = log(ms(r,0));
            }

            dct_o = DCT(log_ms,log_ms.RowNo());

            // ----------- Get to v (Cut row 0,1)--------------
            for (r=0; r<dct_o.RowNo()-2; r++) {
                  v(r,c) = dct_o(r+2,0);
            }
      }
      delete[] re; //--- Don't forget
      delete[] im; //--- Don't forget

      return v;
}
// ------------------------------ DCT -----------------------------
Matrix PROCESS ::DCT(Matrix x,double n) {
      long double sum,angle;
      double k = n;
      Matrix a(x.RowNo(),x.ColNo());

      for (int i=0; i<k; i++) {
            sum = 0;
            for(int j=0; j<n; j++) {
                  angle = ((PI)*((2*(j+1))-1)*((i+1)-1))/(2*n);
                  sum = sum + (x(j,0)*cos(angle));
            }
            if (i==0) {
                  a(i,0) = (1/(sqrt(n)))*sum;
            }
            else {
                  a(i,0) = (sqrt(2/n))*sum;
            }

      }
      return a;
}
// ---------------------------------------------------------------
```

ภาคผนวก จ

ซอร์สโค้ดการเรียกใช้และการเก็บค่าในไฟล์

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <math.h>              //For tanh

int filesize(char *filename);
char * myread(char *filename);
float m[39][27];
char * pEnd;

FILE *stream;

void LoadWBij (void) {
          char* aa;
          string axx,str;
          aa = "WEIGHTij.txt";

          string constructorString (myread(aa));

          int x = 0;
          int y = 0;

          axx = myread("WEIGHTij.txt");

          const char* str2;
          int j=0,k=0;

          for(int s = 0; s < filesize("WEIGHTij.txt"); s++){

             if ( constructorString[s] != '\t' && constructorString[s]
                !='\n' ){x++;}
             else {
                if (constructorString[s] == '\n')       x = x-1;

                str = axx.substr(y,x);
                str2= str.c_str();                      // const char

                float tempMatrix = atof(str2);

                Layers[0].Neurons[k].Dendrites[j].d_weight=tempMatrix;
                y = y+x+1;
                x = 0;
                k++;

                if(k==net_layers[0]){
                   j++; k=0;
                };

                if (constructorString[s] == '\n')       y = y+1;

             }
          }
}
```

```
void ReccordWBij(void){
         stream = fopen( "WEIGHTij.txt", "w+" );
         if( stream == NULL )
            printf( "The file fscanf.out was not opened\n" );
         else{
            for (int j=0;j<net_layers[1];j++){
               for(int k=0;k<net_layers[0];k++){
                  fprintf( stream, "%f",
                        Layers[0].Neurons[k].Dendrites[j].d_weight);
                  if ( k != net_layers[0] -1)
                     fprintf( stream, "%s","\t");
               }
               fprintf( stream, "%s","\n");
            }
            fclose( stream );
         }
}

//************************* Support File ****************************

char * myread(char *filename)
{
      FILE *fp;
      fp = fopen(filename, "rb");
      fseek(fp, 0L, SEEK_END );
      size_t endPos = ftell( fp );

      fseek(fp, 0, SEEK_SET);

      char *tag_buffer;
      tag_buffer = (char*) malloc (endPos);

      fread(tag_buffer, sizeof(char),endPos, fp);
      fclose(fp);
      return tag_buffer;
}

int filesize(char *filename)
{
      FILE *fp;
      fp = fopen(filename, "rb");
      fseek(fp, 0L, SEEK_END );
      size_t endPos = ftell( fp );
      fclose(fp);
      return endPos;
}

//*****************************************************************************
```

ภาคผนวก ฉ

ซอร์สโค้ดการแปลงฟาสท์ฟูริเยร์

```
// ========================================== FFT.H ==========================================//

#define FFT             -1
#define REV_FFT          1

// Radix 2 FFT calculation
// param = FFT     - performs FFT
// param = REV_FFT  - performs reverse FFT
// order = the order of FFT ( pow(2,order) calculated complex points )

void fft(double *real,double *imaginary,int order,int param);
```

```
// ═══════════════════════════ FFT.CPP ═══════════════════════════//

/*
Radix 2 FFT calculation
param = FFT      - performs FFT
param = REV_FFT  - performs reverse FFT
order = the order of FFT ( pow(2,order) calculated complex points )
x = pointer to real
y = pointer to immaginary
*/

#include <math.h>
#include "FFT.H"

#if !defined M_PI
#define M_PI       3.14159265359
#endif

void fft( double *x,double *y,int order,int param ){
        unsigned int n,l,e,f,i,j,o,o1,j1,i1,k;
        double u,v,z,c,s,p,q,r,t,w,a;
        n=1u<<order;
        for( l = 1; l <= order; l++){
                u = 1.0;
                v = 0.0;
                e = 1u << ( order-l+1 );
                f = e/2;
                z = M_PI/f;
                c = cos(z);
                s = sin(z);
                if ( param == FFT ) s = -s;
```

```
for ( j = l; j <= f; j++ ){
        for ( i = j; i <=n; i += e){
                o = i+f-1;
                o1 = i-1;
                p = x[o1]+x[o];
                r = x[o1]-x[o];
                q = y[o1]+y[o];
                t = y[o1]-y[o];
                x[o] = r*u-t*v;
                y[o] = t*u+r*v;
                x[o1] = p;
                y[o1] = q;
        }
        w = u*c-v*s;
        v = v*c+u*s;
        u = w;
    }
}
j = 1;
for ( i = 1; i < n; i++ ){
    if ( i < j ){
        j1 = j-1;
        i1 = i-1;
        p = x[j1];
        q = y[j1];
        x[j1] = x[i1];
        y[j1] = y[i1];
        x[i1] = p;
        y[i1] = q;
    }
    k = n/2;
```

```
            while ( k < j ){
                j = j-k;
                k = k/2;
            }
            j += k;
        }
        if ( param == FFT ) return;

        a = 1.0/n;
        for ( k = 0; k < n; k++ ){
            x[k] *= a;
            y[k] *= a;
        }
        return;
}
```