บทที่ 2
แนวคิดและทฤษฎี

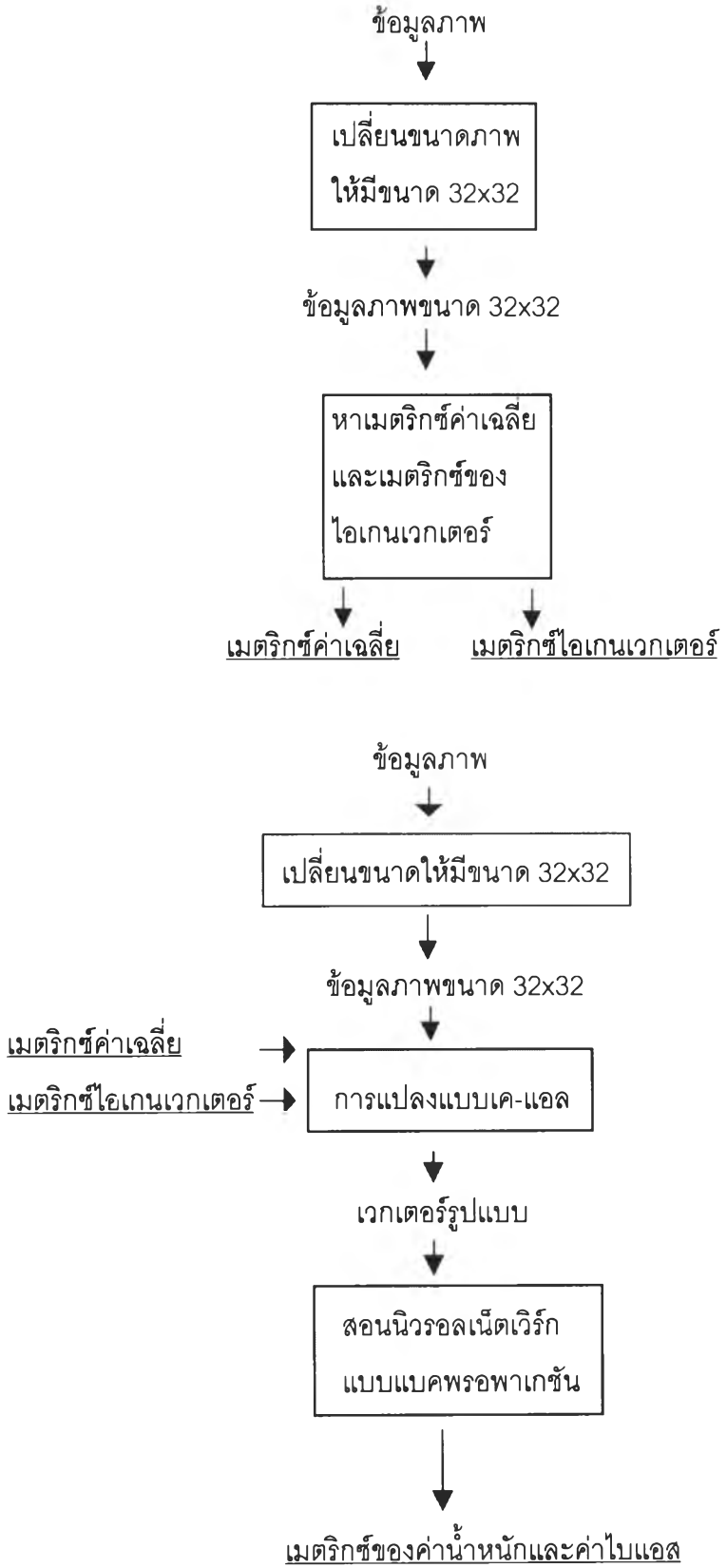
ผังการทำงาน

การรู้จำตัวอักษรพิมพ์ภาษาไทยโดยใช้เทคนิคด้านการวิเคราะห์ตัวประกอบสำคัญ (Principal Component Analysis) และนิวรอลเน็ตเวิร์กสามารถแบ่งหน้าที่การทำงานแบ่งเป็น 2 กระบวนการได้แก่

1. กระบวนการเรียนรู้
2. กระบวนการรู้จำ

กระบวนการเรียนรู้จะเป็นกระบวนการป้อนข้อมูลภาพซึ่งได้จากอุปกรณ์อ่านภาพ (เครื่องสแกนเนอร์) พร้อมกับระบุผลลัพธ์ที่ควรจะได้ ซึ่งข้อมูลภาพทั้งหมดที่ใช้สอนให้เกิดการเรียนรู้จะถูกนำไปคำนวนหาคุณลักษณะ (Feature Extraction) โดยจะหาค่าเมตริกซ์ค่าเฉลี่ย (Mean matrix) และ เมตริกซ์ของไอเกนเวกเตอร์ (Eigenvector) แล้วอาศัยหลักการการแปลงแบบเค-แอล (Karhunen Lo éve Transform, K-L Transform) [8][9][10] แปลงข้อมูลภาพแต่ละภาพเป็นเวกเตอร์รูปแบบ (Pattern vector) เพื่อนำไปทำการสอนนิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชัน (backpropagation) [11][12][13] ผลลัพธ์ของกระบวนการเรียนรู้จะได้เมตริกซ์ค่าเฉลี่ย, เมตริกซ์ของไอเกนเวกเตอร์และเมตริกซ์ของค่าน้ำหนักและค่าไบแอส (bias) ของนิวรอลเน็ตเวิร์ก เมตริกซ์ที่ได้เป็นผลลัพธ์ทั้ง 3 เมตริกซ์นี้จะใช้ในกระบวนการรู้จำต่อไป

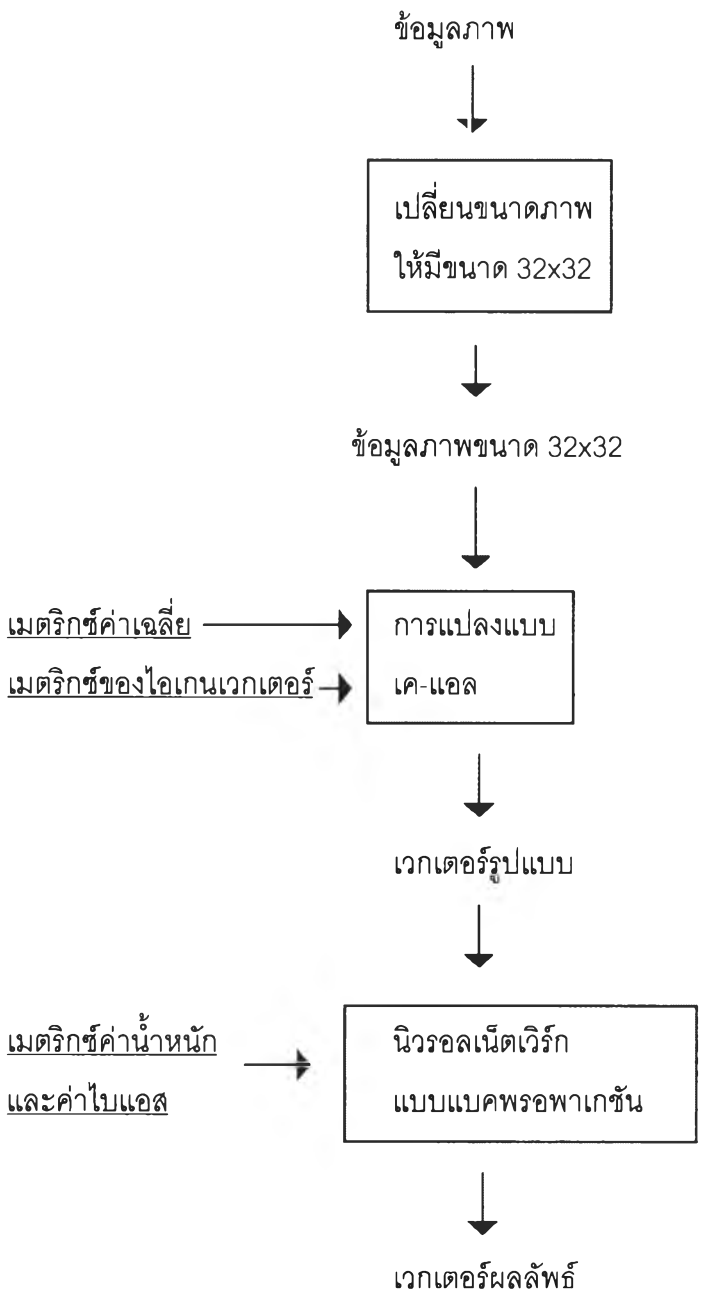
กระบวนการเรียนรู้สามารถแสดงผังการทำงานได้ดังรูปที่ 2.1



รูปที่ 2.1 ผังการทำงานของกระบวนการเรียนรู้

กระบวนการรู้จำจะเป็นกระบวนการที่นำข้อมูลภาพซึ่งได้จากอุปกรณ์อ่านภาพ (เครื่องสแกนเนอร์) มาทำการแปลงแบบเค-แอลเพื่อให้ได้เวกเตอร์รูปแบบ เวกเตอร์รูปแบบจะถูกป้อนสู่นิวรอลเน็ตเวิร์กเพื่อทำการหาเวกเตอร์ผลลัพธ์

กระบวนการรู้จำสามารถแสดงผังการทำงานได้ดังรูปที่ 2.2



รูปที่ 2.2 ผังการทำงานของกระบวนการรู้จำ

ข้อมูลภาพ

ข้อมูลภาพจะรับจากเครื่องสแกนเนอร์โดยจะทำการเก็บภาพแบบขาว-ดำ (bi-level) และจัดในลักษณะของเวกเตอร์ของแต่ละจุด โดยให้แทนค่าจุดด้วย 1 (จุดขาว) และ -1 (จุดดำ) เริ่มจากจุดมุมซ้ายบนของภาพไปยังจุดถัดมาในแถวเดียวกันจนครบทุกจุดบนแถวเดียวกันและเริ่มจุดซ้ายสุดบนภาพของแถวถัดไปดังรูปที่ 2.3

รูปที่ 2.3 ลักษณะการแทนจุดภาพด้วยเวกเตอร์

หลังจากนั้นเวกเตอร์จุดภาพจะถูกเปลี่ยนขนาดเป็น 32x32 (หรือเมตริกซ์ขนาด 1024x1) โดยรักษาอัตราส่วนจำนวนแถวที่มากที่สุดของภาพต่อจำนวนสดมภ์ที่มากที่สุดของจุดภาพเท่าเดิม

การหาเมตริกซ์ค่าเฉลี่ยและเมตริกซ์ของไอเกนเวกเตอร์ (Eigenvector)

เมตริกซ์ค่าเฉลี่ยเป็นเมตริกซ์ที่ได้จากการนำเวกเตอร์ของจุดภาพขนาด 32x32 ที่ได้ทุกภาพของตัวอักษรมาทำการเฉลี่ยได้ตามสมการ

$$\mathbf{m}_x = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i \qquad (1)$$

เมื่อ $\mathbf{x}_i$ คือเวกเตอร์ภาพขนาด 32x32 (เมตริกซ์ขนาด 1024 x 1)

$\mathbf{m}_x$ คือเมตริกซ์ค่าเฉลี่ย (ขนาด 1024 x 1)

M คือจำนวนเวกเตอร์ของจุดภาพทั้งหมดที่เป็นตัวอย่างที่ใช้ขณะเรียนรู้

เมื่อหาเมตริกซ์ค่าเฉลี่ยได้แล้ว จะสามารถหาค่าเมตริกซ์โคแวเรียน (Covariance Matrix) ซึ่งแสดงถึงความแปรผันของข้อมูลภาพทั้งหมดโดยรวมเป็นอย่างไร เมตริกซ์โคแวเรียนหาได้จาก

$$\mathbf{C}_x = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{m}_x\mathbf{m}_x^T \qquad (2)$$

เมื่อ $\mathbf{C}_x$ คือเมตริกซ์โคแวเรียน (ขนาด 1024 x 1024)

การแปลงแบบเค-แอล (K-L Transform)

เนื่องจากเมตริกซ์โคแวเรียน ขนาด 1024 x 1024 มีสมาชิกเป็นจำนวนจริงทั้งหมด และเป็นเมตริกซ์ที่สมมาตร (symmetric) จะสามารถหาไอเกนเวกเตอร์ทั้ง 1024 ตัวได้ (เป็นจำนวนจริง) เสมอ เมื่อนำไอเกนเวกเตอร์ ที่ได้จากเมตริกซ์โคแวเรียน มาสร้างเป็นเมตริกซ์ **A** โดยที่แต่ละแถวของเมตริกซ์ **A** จะประกอบด้วยสมาชิกที่ได้จากไอเกนเวกเตอร์ โดยให้เรียงไอเกนเวกเตอร์ ซึ่งมีค่าของไอเกน (eigenvalue) มากที่สุดอยู่แถวแรก และไอเกนเวกเตอร์ซึ่งมีค่าของไอเกนมากรองมาอยู่แถวถัดไปตามลำดับ จะสามารถทำการแปลงแบบเค-แอลจากข้อมูลภาพที่เปลี่ยนขนาดเป็นขนาด 32x32 แล้วได้จากสมการ

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m}_x) \tag{3}$$

เมื่อ $\mathbf{y}$ คือเวกเตอร์รูปแบบ (เมตริกซ์ขนาด 1024 x 1)

$\mathbf{A}$ คือเมตริกซ์ของการแปลง (ขนาด 1024 x 1024)

$\mathbf{x}$ คือข้อมูลภาพขนาด 32x32 (เมตริกซ์ขนาด 1024 x 1)

$\mathbf{m}_x$ คือเมตริกซ์ของค่าเฉลี่ย (เมตริกซ์ขนาด 1024 x 1)

เนื่องจากการใช้เมตริกซ์ **A** ขนาด 1024 x 1024 จะทำให้เปลืองเนื้อที่หน่วยความจำมาก ถ้าทำการสร้างเมตริกซ์ของการแปลง $\mathbf{A}_k$ จากไอเกนเวกเตอร์เพียง k เวกเตอร์ซึ่งมีค่าของไอเกน (eigenvalue) มากที่สุด โดยให้ค่าสมาชิกในเมตริกซ์ $\mathbf{A}_k$ ในแถวที่ k + 1 ถึงแถวที่ 1024 เป็น 0 จะได้เวกเตอร์รูปแบบซึ่งสมาชิกในแถวที่ k + 1 ถึงแถวที่ 1024 เป็น 0 ซึ่งจะนำเวกเตอร์รูปแบบที่เป็นเมตริกซ์ขนาด k x 1 มาใช้ในขั้นการสอนนิวรอลเน็ตเวิร์กและการรู้จำ การทำเช่นนี้เสมือนว่าข้อมูลภาพขนาด 32x32 จะมีสัญญาณรบกวนซึ่งมีค่าความผิดพลาดกำลังสองเฉลี่ย (mean square error - MSE) เท่ากับค่าของไอเกน (eigenvalue) ตัวที่ไม่ได้ใช้ (ตัวที่ k + 1 ถึงตัวที่ 1024) รวมกัน [8][9] ตามสมการ

$$MSE = \sum_{i=k+1}^{N} l_i \tag{4}$$

เมื่อ N คือจำนวนสมาชิกของเวกเตอร์ภาพขนาด 32x32 (N = 1024)

l_i คือค่าของไอเกน (eigenvalue) ตัวที่ i

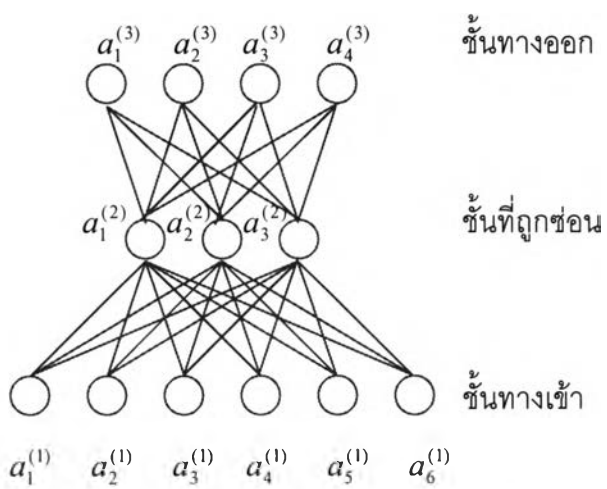
ในขั้นตอนของการเรียนรู้จะเป็นการคำนวนหาค่าเมตริกซ์ของไอเกนเวกเตอร์จากโคแวเรียนเมตริกซ์และเก็บข้อมูลเมตริกซ์ของไอเกนเวกเตอร์ไว้ใช้ในขั้นตอนของการรู้จำต่อไป

ในขั้นตอนของการรู้จำจะเป็นการแปลงแบบเค-แอลของข้อมูลภาพที่ทำการเปลี่ยนขนาดเป็นขนาด 32x32 จุดแล้วโดยการนำค่าเมตริกซ์ของไอเกนเวกเตอร์และเมตริกซ์ของค่าเฉลี่ยที่ได้จากขั้นตอนการรู้จำมาใช้ในการคำนวน

การใช้การแปลงแบบเค-แอลนี้มีแนวคิดว่ากลไกการมองเห็น (perception mechanism) ไม่มีรูปแบบ (model) แต่มีการเชื่อมโยงกับข้อมูลลักษณะเด่นทางสถิติซึ่งประกอบเป็นตัวอักษร ค่าไอเกนเวกเตอร์ของเมตริกซ์ของโคแวเรียนที่ได้จากภาพของตัวอักษรเป็นตัวประกอบสำคัญทางสถิติของความแปรผันจากภาพตัวอักษรต้นแบบ ค่าของไอเกน (eigenvalue) แสดงถึงความสำคัญของไอเกนเวกเตอร์ตัวที่เกี่ยวข้องในความเกี่ยวข้องกับตัวอักษรต้นแบบ ค่าของไอเกน (eigenvalue) ที่น้อยแสดงถึงความไม่เกี่ยวข้องหรือเกี่ยวข้องน้อยมากกับตัวอักษรต้นแบบ[10]

การสอนนิวรอลเน็ตเวิร์ก

นิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชัน (Backpropagation) เป็นนิวรอลเน็ตเวิร์กที่มีชั้นที่ถูกซ่อน (hidden layer) ซึ่งไม่ใช่ชั้นทางเข้า (input layer) และไม่ใช่ชั้นทางออก (output layer) อย่างน้อย 1 ชั้นขึ้นไป นิวรอลเน็ตเวิร์กที่มีชั้นที่ถูกซ่อน 1 ชั้นสามารถแสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 นิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชันที่มีชั้นที่ถูกซ่อน 1 ชั้น

การสอนนิวรอลเน็ตเวิร์กแบบแบคพรอพาเกชันจะเกี่ยวข้องกับงาน 3 ขั้นตอน [13] ได้แก่

1. ขั้นตอนการป้อนรูปแบบที่ต้องการสอนไปข้างหน้า (จากชั้นทางเข้าไปยังชั้นทางออก)
2. ขั้นตอนการคำนวนและกระจายป้อนกลับ (backpropagation) ของค่าความผิดพลาดของผลลัพธ์ (output pattern) กับรูปแบบผลลัพธ์ที่ควรจะเป็น (Target pattern)
3. การปรับค่าน้ำหนัก (weights)

การป้อนเวกเตอร์รูปแบบที่ต้องการสอนไปข้างหน้า

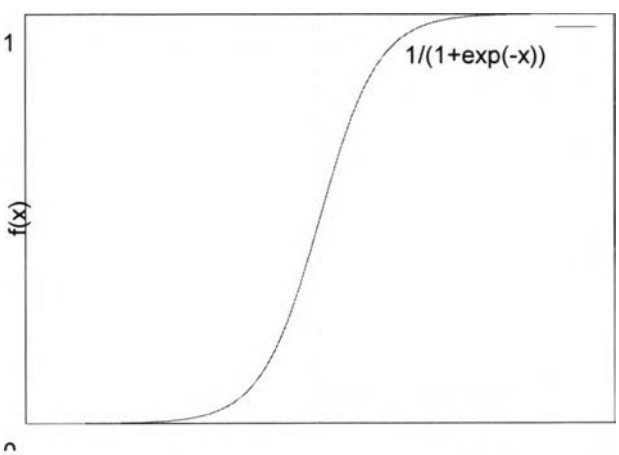
แต่ละบัพ (node) ในแต่ละชั้น (ยกเว้นชั้นทางเข้า) จะให้ค่าผลลัพธ์ $a_i^{(l)}$ จากค่า $u_i^{(l)}$ ที่ป้อนสู่บัพ i ในแต่ละชั้น l ตามสมการ

$$a_i^{(l)} = f(u_i^{(l)}) \tag{5}$$

โดย $f(x)$ เป็นฟังก์ชันกระตุ้น (activation function) ซึ่งมักนิยมใช้ฟังก์ชันซิกมอยด์ (sigmoid function) ซึ่งมีคุณลักษณะตามสมการ

$$f(x) = \frac{1}{1+e^{-\frac{x}{\sigma}}} \tag{6}$$

รูปที่ 2.5 แสดงซิกมอยด์ฟังก์ชันที่มีค่า σ เท่ากับ 1



รูปที่ 2.5 ซิกมอยด์ฟังก์ชัน ที่มีผลลัพธ์ในช่วง (0,1)

คุณลักษณะของค่าที่ป้อนเข้าสู่บัพ i ในแต่ละชั้น l ($u_i^{(l)}$) แสดงได้ตามสมการ

$$u_i^{(l)} = \sum_{j=1}^{N^{(l-1)}} w_{ij}^{(l)} a_j^{(l-1)} + \theta_i^{(l)} \tag{7}$$

โดย $\theta_i^{(l)}$ เป็นค่าไบแอสของบัพ i ในชั้น l

การคำนวนและกระจายป้อนกลับของค่าความผิดพลาดของผลลัพธ์

ค่าความผิดพลาดของผลลัพธ์หาได้ตามสมการ

$$\delta_i^{(l)}(m) \equiv -\frac{\partial \mathrm{E}}{\partial a_i^{(l)}(m)} \tag{8}$$

เมื่อ $\delta_i^{(l)}(m)$ เป็นค่าความผิดพลาดของผลลัพธ์บัพที่ i ที่ชั้น l เมื่อป้อนด้วยเวกเตอร์รูปแบบสำหรับสอนเวกเตอร์ที่ m

และ E เป็นผลต่างกำลังสองที่น้อยที่สุด (least-squares-error) ของเวกเตอร์ผลลัพธ์ที่ได้เมื่อทำการสอนกับเวกเตอร์ที่ควรจะได้ซึ่งหาได้ตามสมการ

$$\mathrm{E} = \frac{1}{2} \sum_{m=1}^{M} \sum_{i=1}^{N} [t_i^{(l)}(m) - a_i^{(l)}(m)]^2 \tag{9}$$

เมื่อ M เป็นจำนวนเวกเตอร์รูปแบบที่ใช้สอน

N เป็นมิติของผลลัพธ์ (จำนวนสมาชิกของเวกเตอร์ผลลัพธ์)

$t_i^{(l)}(m)$ เป็นค่าที่ควรจะได้ที่บัพที่ i ที่ชั้น l เมื่อป้อนด้วยเวกเตอร์รูปแบบสำหรับสอนเวกเตอร์ที่ m

ดังนั้นจะสามารถหาค่าความผิดพลาดของผลลัพธ์ในแต่ละชั้นของนิวรอลเน็ตเวิร์กได้จาก

1. ที่ชั้นผลลัพธ์ ค่าความผิดพลาดของผลลัพธ์ $\delta_i^{(output)}(m)$ หาได้จากสมการ

$$\begin{aligned}\delta_i^{(output)}(m) &= -\frac{\partial \mathrm{E}}{\partial a_i^{(output)}(m)} \\ &= t_i^{(output)}(m) - a_i^{(output)}(m)\end{aligned} \tag{10}$$

2. ที่ชั้น l ใด ๆ ที่ไม่ใช่ชั้นผลลัพธ์

$$\begin{aligned}\delta_i^{(l)}(m) &= -\frac{\partial \mathrm{E}}{\partial a_i^{(l)}(m)} \\ &= -\sum_{j=1}^{N^{(l+1)}} \frac{\partial \mathrm{E}}{\partial u_j^{(l+1)}(m)} \frac{\partial u_j^{(l+1)}(m)}{\partial a_i^{(l)}(m)} \\ &= \sum_{j=1}^{N^{(l+1)}} \delta_j^{(l+1)}(m) f'(u_j^{(l+1)}(m)) w_{ji}^{(l+1)}(m)\end{aligned} \tag{11}$$

การปรับค่าน้ำหนัก

การปรับค่าน้ำหนักจะพยายามทำให้ค่าน้ำหนักที่ได้ (พร้อมกับค่าไบแอส) ทำให้ผลต่างกำลังสองของเวกเตอร์ผลลัพธ์ที่ได้กับเวกเตอร์ผลลัพธ์ที่ควรจะได้มีค่าน้อยที่สุด ซึ่งการปรับค่าน้ำหนักทำได้ตามสมการ

$$w_{ij}^{(l)}(m+1) = w_{ij}^{(l)}(m) + \Delta w_{ij}^{(l)}(m) \tag{12}$$

เมื่อ $w_{ij}^{(l)}(m)$ คือค่าน้ำหนักจากบัพที่ i ในชั้น l ไปยังบัพที่ j ในชั้น $l + 1$ เมื่อใช้เวกเตอร์รูปแบบสำหรับสอนเวกเตอร์ที่ m

และสามารถหาค่า $\Delta w_{ij}^{(l)}(m)$ ได้ตามสมการ

$$\begin{aligned}\Delta w_{ij}^{(l)}(m) &= -\eta \frac{\partial \mathrm{E}}{\partial w_{ij}^{(l)}(m)} \\ &= -\eta \frac{\partial \mathrm{E}}{\partial a_i^{(l)}(m)} f'(u_i^{(l)}(m)) a_j^{(l-1)}(m) \\ &= \eta \delta_i^{(l)}(m) f'(u_i^{(l)}(m)) a_j^{(l-1)}(m) \end{aligned} \quad (13)$$

เมื่อ η คือค่าอัตราการเรียนรู้ (learning rate)

อย่างไรก็ตามในการปรับค่าน้ำหนักมักจะใช้วิธีการคำนวนค่าโมเมนตัมมาร่วมด้วย โดยค่าโมเมนตัมที่นำมาคำนวนนี้จะลดการเกิดออสซิลเลชัน (Oscillation) ซึ่งการปรับน้ำหนักโดยมีค่าโมเมนตัมร่วมด้วยสามารถหาได้ตามสมการ

$$\Delta w_{new} = -\eta \frac{\partial \mathrm{E}}{\partial w_{previous}} + \alpha \Delta w_{previous} \quad (14)$$

เมื่อ η คือค่าอัตราการเรียนรู้ (learning rate)

α คือค่าอินเนอร์เชีย (Inertia)

การใช้นิวรอลเน็ตเวิร์กทำการรู้จำ

หลังจากนิวรอลเน็ตเวิร์กผ่านการเรียนรู้แล้วจะได้ค่าเมตริกซ์ของน้ำหนักและเมตริกซ์ของค่าไบแอส ซึ่งจะถูกนำมาใช้ในการแยกแยะ (classify) เวกเตอร์รูปแบบ (pattern vector) ซึ่งในขั้นการใช้นิวรอลเน็ตเวิร์กทำการรู้จำนี้จะทำงานเพียงขั้นตอนการป้อนเวกเตอร์ที่ต้องการแยกแยะไปข้างหน้า ซึ่งจะใช้วิธีการเดียวกันกับขั้นตอนการป้อนเวกเตอร์ที่ต้องการสอนไปข้างหน้า