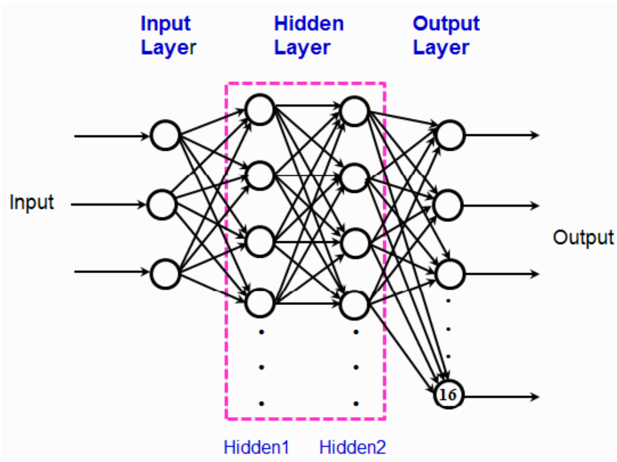
โมเดลของ Multilayer Perceptron (MLP)

เนื่องจากคุณลักษณะของโมเดล Perceptron นั้นเป็นโครงข่ายแบบชั้นเดียวที่มีข้อจำกัดคือ สามารถใช้จำแนกได้เฉพาะรูปแบบข้อมูลที่สามารถแบ่งแยกได้แบบเชิงเส้น (linearly separable) เท่านั้น ในเวลาต่อถึงแม้ว่าจะมีการนำเสนอโมเดล ADALINE (ADAptive LInear NEuron) โดย Marcian Hoff [Widrow and M. E. Hoff, 1960] โดยต่อมาได้นำเสนอโครงข่ายพร้อมกฎการเรียนรู้แบบกำลังสองเฉลี่ยน้อยที่สุดหรือ LMS (Least Mean Square) [Widrow, 1987] ซึ่ง ADALINE นั้นคล้ายคลึงกับ Perceptron แต่ Transfer function ที่ใช้ในโครงข่ายเป็นแบบ “linear” แทนที่จะเป็นแบบ hard limitและการนำกฎการเรียนรู้แบบ LMS มาใช้นั้นทำให้โครงข่ายสามารถทนทานต่อสัญญาณรบกวนได้ดีกว่า Perceptron ซึ่ง ADALINE นั้นมีการนำไปประยุกต์ใช้งานจริงเกี่ยวกับงานทางด้านการประมวลผลสัญญาณดิจิตอลกันมาก อย่างไรก็ตามทั้ง Perceptron และ ADALINE ต่างก็มีข้อจำกัดเดียวกันตรงที่ใช้ได้กับปัญหาที่เป็นแบบแบ่งแยกได้แบบเชิงเส้นเท่านั้น จึงไม่สามารถนำไปประยุกต์ใช้ได้กับปัญหาส่วนใหญ่ได้มากนัก ในเวลาต่อมาแนวคิดเกี่ยวกับ โครงข่ายเพอร์เซฟตรอนแบบหลายชั้น (Multilayer Perceptron: MLP) จึงได้ถูกนำเสนอขึ้น โดยคุณลักษณะพื้นฐานของ MLP คือ [Haykin, 2009]

1. แต่ละนิวรอนในโครงข่ายจะมีการนำ Transfer function ที่เป็นแบบ “nonlinear” มาใช้
2. ชั้น Hidden ที่ไม่ใช่โหนดที่เป็น input และ output มีได้เป็นหนึ่งชั้นหรือมากกว่าก็ได้
3. โครงข่ายมีลักษณะการเชื่อมต่อกันสูง (high degree of connectivity) ด้วยค่าของ Synaptic weight

จากคุณลักษณะพื้นฐานดังกล่าวของ MLP จึงสามารถนำไปใช้ประยุกต์ใช้ได้กับงานในหลายๆ ด้าน จึงเป็นที่รู้จักและเป็นที่นิยมในเวลาต่อมา ซึ่งโครงข่าย MLP นั้นเป็นสถาปัตยกรรมแบบป้อนไปข้างหน้า (Feedforward) ที่ประกอบด้วยชั้นต่างๆ 3 ชั้นหลักคือ ชั้นข้อมูลเข้า (Input Layer) ชั้นซ่อนเร้น (Hidden Layer) และชั้นเอาต์พุต (Output Layer) โดยชั้นซ่อนเร้นอาจมีได้มากกว่าหนึ่งชั้น ดังแสดงในรูปที่ 10 ซึ่งเป็นตัวอย่างของโครงข่าย ที่มีชั้นซ่อนเร้นสองชั้น (Hidden1 และ Hidden2) โดยแต่ละชั้นซ่อนเร้นนั้นอาจมีจำนวนโหนดหรือจำนวนนิวรอน (Neuron) หนึ่งนิวรอนหรือมากกว่าได้ขึ้นอยู่การนำไปประยุกต์ใช้กับแต่ละปัญหา จากรูปที่ 1 เป็นตัวอย่างที่มีจำนวนนิวรอนในชั้นอินพุตเป็น 3 นิวรอน และมีจำนวนนิวรอนในชั้นเอาต์พุตเป็น 16 นิวรอน



รูปที่ 1. สถาปัตยกรรมของโครงข่ายประสาทเทียมแบบ MLP

โครงข่าย MLP ที่มีการนำเทคนิคการเรียนรู้หรืออัลกอรึทึมแบบแพร่กลับ (Back Propagation) ที่นำเสนอโดย Rumelhart [Rumelhart, et al., 1986a, 1986b] มีการนำมาประยุกต์ใช้อย่างกว้างขวางในหลายๆ ด้านในเวลาต่อมา ไม่ว่าจะเป็นการรู้จำรูปแบบ (Pattern Recognition) การจำแนก (Classification) ข้อมูล รวมถึงการประมาณค่า (Approximation) และการพยากรณ์ (Forecasting) การทำงานของอัลกอรึทึมแบบแพร่กลับ ทีมี่จำนวนนิวรอนในชั้นอินพุต ชั้นซ่อนเร้น และชั้นเอาต์พุต เป็น n, q และ p ตามลำดับ มีขั้นตอนดังต่อไปนี้คือ [Kumar, 2005].

อัลกอรึทึมแบบแพร่กลับ *(Backpropagation Algorithm)*

กำหนดให้:

- ข้อมูลแต่ละตัวที่ต้องการฝึกสอนคือเวกเตอร์ $X_{\mathrm{k}} \in \mathbb{R}^n$
- เอาต์พุตเป้าหมาย (Target Output) ของข้อมูลแต่ละตัวคือเวกเตอร์ $D_{\mathrm{k}} \in \mathbb{R}^p$

กำหนดค่าเริ่มต้น:

- กำหนดค่า weight เริ่มต้นแบบสุ่มให้กับ weight ทุกตัวที่เชื่อมระหว่างขั้นอินพุตและชั้นซ่อนเร้น นั่นคือ w_{ih}^1 และให้ $\Delta w_{ih}^0 = 0,\ i = 0, \dots, n;\ h = 1, \dots, q$
- กำหนดค่า weight เริ่มต้นแบบสุ่มให้กับ weight ทุกตัวที่เชื่อมระหว่างชั้นซ่อนเร้นและชั้นเอาต์พุต ซึ่งคือ w_{hj}^1 และให้ $\Delta w_{hj}^0 = 0,\ h = 0, \dots, q;\ j = 1, \dots, p$
- ให้ $k = 1$ และกำหนดค่าโมเมนตัม η ค่าคงที่การเรียนรู้ α และค่าความผิดพลาดที่ยอมรับได้ τ ตามที่ต้องการ

วนลูปเพื่อทำซ้ำในแต่ละ k:

{

- เลือกข้อมูลมาหนึ่งคู่ นั่นคือข้อมูลที่ต้องการฝึกสอนและเอาต์พุตเป้าหมายที่ต้องการของข้อมูลตัวนั้น กำหนดให้เป็น (X_k, D_k)
- คำนวณค่าสัญญาณต่างๆ ในขั้นตอน forward ตามลำดับของสมการต่อไปนี้

$$\mathcal{S}(x_i^k) = x_i^k, \qquad i = 1, \dots, n$$

$$\mathcal{S}(x_0^k) = 1$$

$$\mathcal{Z}_h^k = \sum_{i=0}^{n} w_{ih}^k x_i^k, \qquad h = 1, \dots, q$$

$$\mathcal{S}(\mathcal{Z}_h^k) = \frac{1}{1 + \exp(-\mathcal{Z}_n^k)}, \qquad h = 1, \dots, q$$

$$\mathcal{S}(\mathcal{Z}_0^k) = 1$$

$$y_j^k = \sum_{h=0}^{q} w_{hj}^k \mathcal{S}(\mathcal{Z}_h^k), \qquad j = 1, \dots, p$$

$$\mathcal{S}(y_j^k) = \frac{1}{1 + \exp(-y_j^k)}, \qquad j = 1, \dots, p$$

– คำนวณ delta/error ที่นิวรอนเอาต์พุต และหาค่า weight ที่เปลี่ยนแปลงไประหว่างชั้นซ่อนเร้นและชั้นเอาต์พุต ตามสมการคือ

$$\delta_j^k = \left(d_j^k - \mathcal{S}(y_j^k)\right)\mathcal{S}'(y_j^k), \qquad j = 1, \dots, p$$

โดยที่ $S'(y_j^k) = \mathcal{S}(y_j^k)(1 - \mathcal{S}(y_j^k))$

$$\Delta w_{hj}^k = \eta \delta_j^k \mathcal{S}(\mathcal{Z}_h^k) \qquad h = 0, \dots, q; j = 1, \dots, p$$

– คำนวณ delta/error ที่นิวรอนชั้นซ่อนเร้น ตามสมการคือ

$$\delta_h^k = \left(\sum_{j=1}^{p} \delta_j^k w_{hj}^k\right)\mathcal{S}'(\mathcal{Z}_h^k) \qquad h = 1, \dots, q$$

$$\Delta w_{ih}^k = \eta \delta_h^k x_i^k \qquad i = 0, \dots, n; h = 1, \dots, q$$

– ปรับค่าของ weight ตามสมการคือ

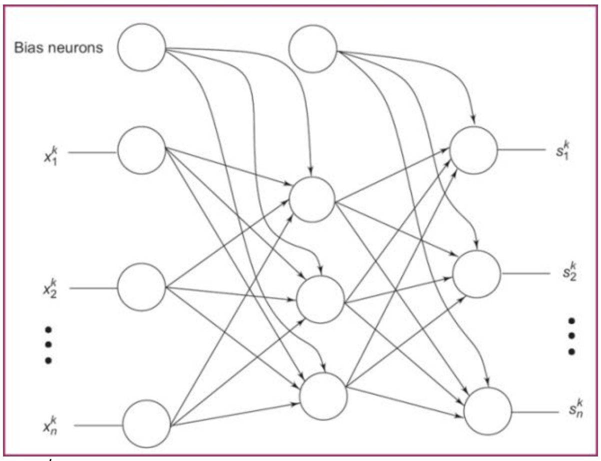
$$w_{hj}^{k+1} = w_{hj}^k + \Delta w_{hj}^k + \alpha \Delta w_{hj}^{k-1} \qquad h = 0, \dots, q; j = 1, \dots, p$$

$$w_{ih}^{k+1} = w_{ih}^k + \Delta w_{ih}^k + \alpha \Delta w_{ih}^{k-1} \qquad i = 0, \dots, n; h = 1, \dots, q$$

– คำนวณค่าความผิดพลาดให้เป็น $\varepsilon_k = \frac{1}{2}\sum_{i=1}^{p}(d_j^k - s(y_j^k))^2$

} ทำจนกระทั่ง ($\varepsilon_{av} = \frac{1}{Q}\sum_{k=1}^{Q} \varepsilon_k < \tau$)
เมื่อ Q คือจำนวนข้อมูล

จากขั้นตอนของ algorithm ดังกล่าว เพื่อให้เข้าใจการทำงานชัดเจนมากยิ่งขึ้น จึงยกตัวอย่างการคำนวณด้วยมือ (hand work example) ที่แสดงรายละเอียดของค่าต่างๆ ที่เกิดขึ้นในแต่ละขั้นตอน โดยเป็นการยกตัวอย่างที่สอดคล้องกับโครงข่ายดังรูปที่ 2 ที่มองภาพของค่า bias เป็นเสมือนนิวรอนหนึ่งในเครือข่ายด้วย



รูปที่ 2. โครงข่าย MLP ที่ใช้ประกอบการยกตัวอย่างการคำนวณด้วยมือ (hand work example)

ตารางที่ 1. แสดงสัญลักษณ์ต่างๆ ที่ใช้ใน Backpropagation algorithm ซึ่งสอดคล้องกับโครงข่ายใน รูปที่ 2

ตารางที่ 1. สัญลักษณ์ต่างๆ ที่ใช้ใน Backpropagation algorithm

	Input		*Hidden*		*Output*
Number of neurons	$n+1$		$q+1$		p
Signal function	linear		sigmoidal		sigmoidal
Neuron index range	$i = 0, \ldots, n$		$h = 0, \ldots, q$		$j = 1, \ldots, p$
Activation	x_i		z_h		y_j
Signal	$\mathcal{S}(x_i)$		$\mathcal{S}(z_h)$		$\mathcal{S}(y_j)$
Weights (including bias)		$\rightarrow w_{ih} \rightarrow$		$\rightarrow w_{hj} \rightarrow$	

สำหรับโครงข่ายที่ใช้ยกตัวอย่างนั้นมี Architecture แบบ ***n-p-q*** นั่นคือ เป็นโครงข่ายที่มี hidden layer เพียงชั้นเดียว ที่มี ***n=2, p= 2 และ q=2*** (จำนวน neuron ในชั้น Input มี 2 นิรอน, ชั้น Hidden มี 2 นิรอน, และ ชั้น Output มี 2 นิรอนเช่นกัน) สำหรับค่าในตารางที่ 1 ที่แสดงค่าของ Number of neurons ในชั้น Input เป็น n+1 เพราะเป็นการมองค่าของ bias เป็นอีกนิวรอนหนึ่งด้วย เช่นเดียวกันกับในชั้น Hidden ที่เป็น q+1

ตารางที่ 2 เป็นข้อมูลสองตัว (สอง Pattern Index) ที่ใช้ประกอบการยกตัวอย่างโดยแต่ละตัวมีสอง elements คือ (x_1, x_2) ส่วน (d_1, d_2) เป็น target output ที่สัมพันธ์กัน ซึ่งสอดคล้องกับขั้นตอนใน algorithm คือ

กำหนดให้:

- ข้อมูลแต่ละตัวที่ต้องการฝึกสอนคือเวกเตอร์ $X_k \in \mathbb{R}^n$
- เอาต์พุตเป้าหมาย (Target Output) ของข้อมูลแต่ละตัวคือเวกเตอร์ $D_k \in \mathbb{R}^p$

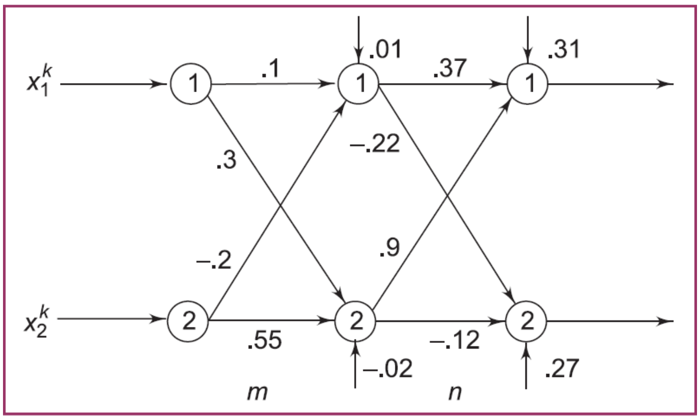
ตารางที่ 2. ข้อมูลที่ใช้ประกอบการยกตัวอย่าง

Pattern Index	x_1^k	x_2^k	d_1^k	d_2^k
1	0.5	–0.5	0.9	0.1
2	–0.5	0.5	0.1	0.9

รูปที่ 3 แสดงค่าของ weight เริ่มต้นที่กำหนดให้กับโครงข่าย เมื่อ k=1 นั่นคือกำหนดค่าแบบสุ่มให้กับ w_{ih}^1 และ w_{hj}^1 แต่ละค่าแสดงในรายละเอียดของเส้นเชื่อมค่า weight ดังรูปที่ 3 ซึ่งสอดคล้องกับขั้นตอนใน algorithm คือ

กำหนดค่าเริ่มต้น:

- กำหนดค่า weight เริ่มต้นแบบสุ่มให้กับ weight ทุกตัวที่เชื่อมระหว่างชั้นอินพุตและชั้นซ่อนเร้น นั่นคือ w_{ih}^1 และให้ $\Delta w_{ih}^0 = 0,\ i = 0, \ldots, n;\ h = 1, \ldots, q$
- กำหนดค่า weight เริ่มต้นแบบสุ่มให้กับ weight ทุกตัวที่เชื่อมระหว่างชั้นซ่อนเร้นและชั้นเอาต์พุต ซึ่งคือ w_{hj}^1 และให้ $\Delta w_{hj}^0 = 0,\ h = 0, \ldots, q;\ j = 1, \ldots, p$
- ให้ $k = 1$ และกำหนดค่าคงที่การเรียนรู้ $\eta = 1.2$ ค่าโมเมนตัม $\alpha = 0.8$ และค่าความผิดพลาดที่ยอมรับได้ τ ตามที่ต้องการ



รูปที่ 3. ค่า weight เริ่มต้นแบบสุ่มที่กำหนดให้กับ w_{ih}^1 และ w_{hj}^1

เมื่อ k=1 จะพิจารณา Pattern Index ที่ 1 ของตารางที่ 2 แล้วคำนวณค่าสัญญาณต่างๆ ในขั้นตอน forward ของ algorithm นั่นคือเป็นการคำนวณตามลำดับของสมการดังตารางด้านล่าง

เมื่อ $k = 1$:

- เลือกข้อมูลมาหนึ่งคู่ นั่นคือข้อมูลที่ต้องการฝึกสอนและเอาต์พุตเป้าหมายที่ต้องการของข้อมูลตัวนั้น กำหนดให้เป็น (X_1, D_1) นั่นคือ พิจารณา Pattern Index ที่ 1 ของตารางที่ 2
- คำนวณค่าสัญญาณต่างๆ ในขั้นตอน forward ตามลำดับของสมการต่อไปนี้

$$\mathcal{S}(x_i^k) = x_i^k, \qquad i = 1, \ldots, n$$

$$\mathcal{S}(x_0^k) = 1$$

$$\mathcal{Z}_h^k = \sum_{i=0}^{n} w_{ih}^k x_i^k, \qquad h = 1, \ldots, q$$

$$\mathcal{S}(\mathcal{Z}_h^k) = \frac{1}{1 + \exp(-\mathcal{Z}_n^k)}\ , \qquad h = 1, \ldots, q$$

$$\mathcal{S}(\mathcal{Z}_0^k) = 1$$

$$y_j^k = \sum_{h=0}^{q} w_{hj}^k \mathcal{S}(z_h^k), \qquad j = 1, \dots, p$$

$$\mathcal{S}(y_j^k) = \frac{1}{1 + \exp(-y_j^k)}, \qquad j = 1, \dots, p$$

นั่นคือ ณ k=1, เมื่อพิจารณา Pattern Index ที่ 1 ซึ่งคือ (x_1, x_2) = (0.5, -0.5) ส่วน (d_1, d_2) = (0.9, 0.1) แล้วนำมาแทนค่าคำนวณค่าต่างๆ ทั้งหมด ในขั้นตอนforward จะได้ผลลัพธ์ค่าสัญญาณต่างๆ ดังตารางที่ 3 ด้านล่าง

ตารางที่ 3. ค่าต่างๆ ที่ได้จากการคำนวณในขั้นตอน forward เมื่อ k=1

k	x_1^k	x_2^k	$\mathcal{S}(x_1^k)$	$\mathcal{S}(x_2^k)$	z_1^k	z_2^k	$\mathcal{S}(z_1^k)$	$\mathcal{S}(z_2^k)$	y_1^k	y_2^k	$\mathcal{S}(y_1^k)$	$\mathcal{S}(y_2^k)$
1	.5	–.5	.5	–.5	.16	–.145	.5399	.4638	.9271	.0955	.7164	.5238

ขั้นตอนต่อมาเป็นการคำนวณ delta/error ที่นิวรอนชั้นเอาต์พุต และคำนวณ delta/error ที่นิวรอนชั้นซ่อนเร้น ตามลำดับดังสมการด้านล่าง

- คำนวณ delta/error ที่นิวรอนชั้นเอาต์พุต และหาค่า weight ที่เปลี่ยนแปลงไประหว่างชั้นซ่อนเร้นและชั้นเอ๊าพุต ตามสมการคือ

$$\delta_j^k = \left(d_j^k - \mathcal{S}(y_j^k)\right)\mathcal{S}'(y_j^k) \qquad j = 1, \dots, p$$

โดยที่ $\mathcal{S}'(y_j^k) = \mathcal{S}(y_j^k)(1 - \mathcal{S}(y_j^k))$

$$\Delta w_{hj}^k = \eta \delta_j^k \mathcal{S}(z_h^k) \qquad h = 0, \dots, q; j = 1, \dots, p$$

- คำนวณ delta/error ที่นิวรอนชั้นซ่อนเร้น และหาค่า weight ที่เปลี่ยนแปลงไประหว่างชั้นอินพุตและชั้นซ่อนเร้น ตามสมการคือ

$$\delta_h^k = \left(\sum_{j=1}^{p} \delta_j^k w_{hj}^k\right)\mathcal{S}'(z_h^k) \qquad h = 1, \dots, q$$

โดยที่ $\mathcal{S}'(z_h^k) = \mathcal{S}(z_h^k)(1 - \mathcal{S}(z_h^k))$

$$\Delta w_{ih}^k = \eta \delta_h^k x_i^k \qquad i = 0, \dots, n; h = 1, \dots, q$$

ค่าของ δ_1^1 ด้านล่างเป็นค่า delta/error ของนิวรอนตัวที่ 1 ในชั้นเอาต์พุต และ δ_2^1 เป็นค่า delta/error ของนิวรอนตัวที่ 2 ในชั้นเอาต์พุตเช่นกัน

$$e_1^1 = d_1^1 - s_1^1 = 0.9 - 0.7164 = 0.1836$$
$$e_2^1 = d_2^1 - s_2^1 = 0.1 - 0.5238 = -0.4238$$
$$\delta_1^1 = 0.1836 \times 0.7164(1 - 0.7164) = 0.0373$$
$$\delta_2^1 = -0.4238 \times 0.5238(1 - 0.5238) = -0.1057$$

ค่า weight ที่เปลี่ยนแปลงไประหว่างชั้นซ่อนเร้นและชั้นเอ๊าพุตได้ผลลัพธ์ดังด้านล่าง ในที่นี้ เพื่อไม่ให้สับสนกับ parameter ต่างๆ จึงกำหนดให้ Δw_{hj}^{k} ใน algorithm แทนด้วย Δn_{hj}^{k}

$$\Delta n_{01}^{1} = 1.2 \times 0.0373 \times 1.0 = 0.0447$$
$$\Delta n_{11}^{1} = 1.2 \times 0.0373 \times 0.5399 = 0.0241$$
$$\Delta n_{21}^{1} = 1.2 \times 0.0373 \times 0.4638 = 0.0207$$
$$\Delta n_{02}^{1} = 1.2 \times -0.1057 \times 1.0 = -0.1268$$
$$\Delta n_{12}^{1} = 1.2 \times -0.1057 \times 0.5399 = -0.0684$$
$$\Delta n_{22}^{1} = 1.2 \times -0.1057 \times 0.4638 = -0.0588$$

ส่วนค่าของ δH_1^1 ด้านล่างเป็นค่า delta/error ของนิวรอนตัวที่ 1 ในชั้นซ่อนเร้น และ ของ δH_2^1 delta/error ของนิวรอนตัวที่ 2 ในชั้นซ่อนเร้นเช่นกัน

$$\delta H_1^1 = \left(0.0373 \times 0.37 + (-0.1057 \times -0.22)\right) \times 0.5399(1 - 0.5399) = 0.0092$$
$$\delta H_2^1 = \left(0.0373 \times 0.9 + (-0.1057 \times -0.12)\right) \times 0.4638(1 - 0.4638) = 0.0115$$

ค่า weight ที่เปลี่ยนแปลงไประหว่างชั้นอินพุตและชั้นซ่อนเร้นได้ผลลัพธ์ดังด้านล่าง ในที่นี้ เพื่อไม่ให้สับสนกับ parameter ต่างๆ จึงกำหนดให้ Δw_{ih}^{k} ใน algorithm แทนด้วย Δm_{ih}^{k}

$$\Delta m_{01}^{1} = 1.2 \times 0.0092 \times 1.0 = 0.011$$
$$\Delta m_{11}^{1} = 1.2 \times 0.0092 \times 0.5 = 0.0055$$
$$\Delta m_{21}^{1} = 1.2 \times 0.0092 \times -0.5 = -0.0055$$
$$\Delta m_{02}^{1} = 1.2 \times 0.0115 \times 1.0 = 0.0138$$
$$\Delta m_{12}^{1} = 1.2 \times 0.0115 \times 0.5 = 0.0069$$
$$\Delta m_{22}^{1} = 1.2 \times 0.0115 \times -0.5 = -0.0069$$

ขั้นตอนถัดมา คือการปรับค่า weight ตามสมการด้านล่าง

- ปรับค่าของ weight ตามสมการคือ

$$w_{hj}^{k+1} = w_{hj}^{k} + \Delta w_{hj}^{k} + \alpha \Delta w_{hj}^{k-1} \qquad h = 0, \dots, q; j = 1, \dots, p$$

$$w_{ih}^{k+1} = w_{ih}^{k} + \Delta w_{ih}^{k} + \alpha \Delta w_{ih}^{k-1} \qquad i = 0, \dots, n; h = 1, \dots, q$$

ผลลัพธ์ที่ได้จากการปรับค่า weight แล้วได้เป็นค่าของ weight ใหม่ด้านล่าง เมื่อ n_{hj}^{2} คือ weight ที่เชื่อมระหว่างชั้นซ่อนเร้นกับชั้นเอ๊าพุต ส่วน m_{ih}^{2} คือ weight ที่เชื่อมระหว่างอินพุตกับชั้นซ่อนเร้น

$n_{01}^2 = 0.31 + 0.0447 = 0.3547$ $\quad$ $m_{01}^2 = 0.01 + 0.011 = 0.021$

$n_{11}^2 = 0.37 + 0.0241 = 0.3941$ $\quad$ $m_{11}^2 = 0.1 + 0.0055 = 0.1055$

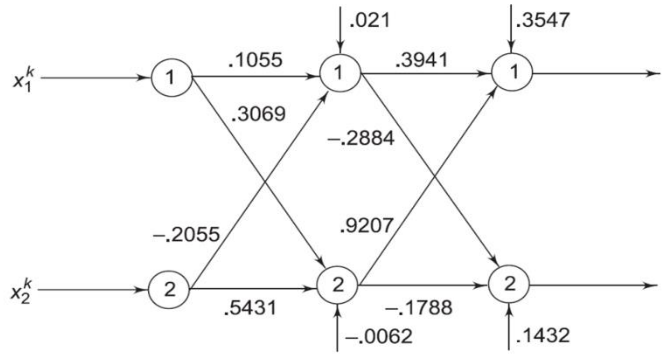
$n_{21}^2 = 0.9 + 0.0207 = 0.9207$ $\quad$ $m_{21}^2 = -0.2 - 0.0055 = -0.2055$

$n_{02}^2 = 0.27 - 0.1268 = 0.1432$ $\quad$ $m_{02}^2 = -0.02 + 0.0138 = -0.0062$

$n_{12}^2 = -0.22 - 0.0684 = -0.2884$ $\quad$ $m_{12}^2 = 0.3 + 0.0069 = 0.3069$

$n_{22}^2 = -0.12 - 0.0588 = -0.1788$ $\quad$ $m_{22}^2 = 0.55 - 0.0069 = 0.5431$

รูปที่ 4 แสดงค่าของ weight สุดท้ายแต่ละค่าบนโครงข่าย หลังจบขั้นตอนของการรับข้อมูลตัวแรก นั่นคือเป็นการนำค่าของ m_{ih}^2 และ n_{hj}^2 ด้านบนมาใส่ในแต่ละเส้นเชื่อมของค่า weight



รูปที่ 4. ค่า weight สุดท้ายทั้งหมดที่ได้ของโครงข่าย หลังรับข้อมูลตัวแรก

ขั้นตอนสุดท้ายของการทำงานเมื่อ k=1 คือการหาค่าความผิดพลาดของ Pattern Index ที่รับเข้ามาดังสมการด้านล่าง

คำนวณค่าความผิดพลาดให้เป็น $\varepsilon_k = \frac{1}{2}\sum_{i=1}^{p}(d_j^k - s(y_j^k))^2$

จากนั้นโครงข่ายก็จะพิจารณา Pattern Index ถัดไป โดยทำขั้นตอนต่างๆ ทั้งหมดตามลำดับเช่นเดียวกันกับที่ยกตัวอย่างสำหรับ Pattern Index ณ k=1 ก่อนหน้านี้ จนจบของแต่ละ k นั่นคือ ถ้าข้อมูลมีทั้งหมด 100 ตัวก็จะทำจนจบ k=100 ซึ่งคือใน 1 epoch (1 รอบใหญ่) จะพิจารณาข้อมูลทุกตัวจนครบแล้วหาค่าความผิดพลาดรวมตามสมการคือ

$$(\varepsilon_{av} = \frac{1}{Q}\sum_{k=1}^{Q}\varepsilon_k < \tau)$$

เมื่อ *Q* คือจำนวนข้อมูล

นั่นคือ ถ้าค่าความผิดพลาดรวมยังมากกว่าค่าที่ยอมรับได้ (τ) ที่ถูกกำหนดไว้ก็จำทำต่อไปครั้งละ 1 epoch ไปเรื่อยๆ โดยค่า τ มักกำหนดเป็นค่าน้อยๆ เช่น 0.01, 0.001 เป็นต้น