การเพิ่มความเร็ว SOM ในการจัดกลุ่มข้อมูลด้วยเทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นและความเป็นปึกแผ่นของข้อมูล

Increasing speed SOM for Clustering Data with A Density-Biased Sampling and Partial Approximate Compactness Estimator

บัณฑิต ปุญญวัฒนะ บุญวัฒน์ อัตชู

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทคัดย่อ

Self-Organizing Map(SOM) เป็นนิวรอนเน็ตเวิร์กแบบไม่มีผู้สอน จุดเด่นคือเหมาะสำหรับนำมาวิเคราะห์ข้อมูล โดยข้อมูลที่ได้จะอยู่ในรูปแบบของกริดที่แสดงเป็นแผนภาพช่วยในการวิเคราะห์ข้อมูลได้เป็นอย่างดี จุดอ่อนของแผนภาพ SOM ก็คือจะใช้ระยะเวลาในการเรียนรู้นานเมื่อแผนภาพมีขนาดใหญ่และข้อมูลมีปริมาณมาก ซึ่งการลดขนาดของข้อมูลที่ใช้เรียนรู้ของ SOM เป็นอีกแนวทางหนึ่งที่จะช่วยแก้ปัญหานี้ได้ งานวิจัยนี้จึงนำวิธีการลดขนาดข้อมูลด้วยเทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นและความเป็นปึกแผ่นของข้อมูลซึ่งสามารถสร้างชุดข้อมูลสุ่มแบบต่อเนื่องได้โดยการอ่านข้อมูลเพียงรอบเดียวแล้วนำข้อมูลที่ได้จากการสุ่มไปทำการเรียนรู้ในแผนภาพของ SOM เพื่อให้แผนภาพมีการจัดเรียงตัวได้เร็วขึ้นแล้วจึงนำข้อมูลทั้งหมดมาเรียนรู้อีกครั้ง พบว่าวิธีที่นำเสนอสามารถลดเวลาในการจัดข้อมูลได้มากกว่า 50 เปอร์เซ็นต์เมื่อเปรียบเทียบกับ SOM แบบเดิมและประสิทธิภาพของการจัดกลุ่มเอกสารยังคงเดิม

คำสำคัญ : เซลฟ์ออร์แกไนสซิ่งแม็พ, การสุ่มแบบเบี่ยงเบนตามความหนาแน่น, ระยะทางแบบยูคลิด

Abstract

Self-Organizing Map(SOM) is an unsupervised neural network providing cluster analysis of high dimensional input data. Outputs from SOM are represented in map that help us to explore data. The weak point of conventional SOM is that it takes long time to train system in case of large map and large data set. Data reduction is an important step to increase the efficiency of SOM. This research presents A Density-Biased Sampling and Partial Approximate Compactness Estimator (DBSPACE) to reduce data. This method requires only one pass of dataset. The output then is used for training SOM to facilitate fast arrangement. Lastly, the learning process is repeated with the whole data. The result from experiment shows that this method can reduce time up to 50 percent, compared with the conventional SOM, yet still get the same result to find cluster.

Key words: Self-Organizing Map, A Density-Biased Sampling , Euclidean distance

1.บทนำ

SOM ถูกนำมาใช้อย่างกว้างขวางในหลายๆด้านเช่น การประยุกต์ใช้ SOM ในการจดจำเสียง[1] การประยุกต์ใช้ SOM ในการจัดกลุ่มข่าวบนอินเตอร์เน็ต[2] เป็นต้น นอกจากนี้แล้วยังมีงานวิจัยอื่นๆเกี่ยวกับ SOM ได้รับการตีพิมพ์มากกว่า 3000 บทความในช่วงปี ค.ศ.1981-1997[3] อย่างไรก็ตามปัญหาสำคัญในการประยุกต์ใช้งาน SOM คือ เวลาในการเรียนรู้ ซึ่งจะใช้เวลาในการเรียนรู้ค่อนข้างนาน ถ้าข้อมูลมีมิติสูง เช่น เวคเตอร์ของเอกสารมีขนาดมากจะยิ่งใช้เวลาในการเรียนรู้นาน จึงมีงานวิจัยที่พัฒนาลดเวลาในการเรียนรู้ เช่น การนำเอาอัลกอริทึม K-mean ช่วยในการจัดกลุ่มข้อมูลก่อนนำไปเรียนรู้ในแผนภาพเพื่อให้แผนภาพเรียงตัวได้เร็วขึ้นในจำนวนรอบที่น้อยกว่า[4] การลดการคำนวณโหนดชนะโดยแบ่งเป็นส่วนย่อยๆ ที่ใช้ในการค้นหาวิธีนี้สามารถลดเวลาในการคำนวณได้สูงสุดถึง 14 เปอร์เซ็นต์[5] งานวิจัยนี้นำวิธีการลดขนาดข้อมูลด้วยเทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นและความเป็นปึกแผ่นของข้อมูลแล้วนำข้อมูลที่ได้จากการสุ่มไปทำการเรียนรู้ในแผนภาพของ SOM เพื่อแก้ปัญหาการใช้เวลาในการเรียนรู้นานโดยประสิทธิภาพการจัดกลุ่มยังคงเดิม

2.ทฤษฎีที่เกี่ยวข้อง

2.1 เทคนิคการสุ่มแบบเบี่ยงเบนตามความหนาแน่นและความเป็นปึกแผ่นของข้อมูล (DBSPACE: A Density-Biased Sampling and Partial Approximate Compactness Estimator)

DBSPACE [6]เป็นเทคนิคที่ใช้สำหรับงานการจัดกลุ่มข้อมูลขนาดใหญ่ ลักษณะที่สำคัญของอัลกอริทึมคือ สามารถสร้างชุดข้อมูลสุ่มได้ด้วยการอ่านข้อมูลเพียงแค่รอบเดียวเท่านั้น, รองรับข้อมูลที่มีขนาดใหญ่มากและไม่สามารถทราบจำนวนข้อมูลทั้งหมดมาก่อนล่วงหน้าการสุ่มข้อมูล, สร้างชุดข้อมูลสุ่มที่ดีสำหรับข้อมูลที่มีการกระจายแบบไม่ปกติ (Zipf), สามารถตรวจสอบและกรองข้อมูลรบกวนไม่ให้ถูกเลือกเข้ามาในชุดข้อมูลสุ่มโดยใช้ค่าประมาณความเป็นปึกแผ่นของข้อมูลประกอบการสุ่มข้อมูล การคำนวณค่าต่างๆที่จำเป็นสำหรับอัลกอริทึมมีดังนี้ ให้ $\{\vec{x}_i\}$ เป็นเซ็ตข้อมูลขนาด d มิติจำนวน N อ็อบเจกต์ของคลัสเตอร์ใดๆ บน data space โดยที่ i =1,2,…, N

Centroid ($\vec{\mu}$) หรือจุดศูนย์กลางมวลของคลัสเตอร์เป็นจุดที่ใช้เป็นตัวแทนของทุกอ็อบเจกต์ในคลัสเตอร์ หาได้ดังสมการที่ (1)

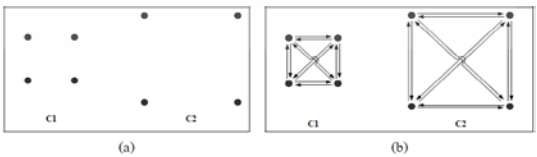
$$\vec{\mu} = \frac{\sum_{i=1}^{N} \vec{x}_i}{N} \qquad (1)$$

Diameter (D) หรือเส้นผ่านศูนย์กลางของคลัสเตอร์เป็นค่าระยะทางเฉลี่ยของทุกคู่อ็อบเจกต์ภายในคลัสเตอร์เดียวกัน ให้ $d(\vec{x}_i, \vec{x}_j)$ เป็นระยะทางตามแบบยูคลิด (Euclidean distance) ระหว่างอ็อบเจกต์ $\vec{x}_i$ กับ $\vec{x}_j$ diameter ของคลัสเตอร์สามารถหาได้จาก

$$D = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} d(\vec{x}_i, \vec{x}_j)^2}{N(N-1)}} \qquad (2)$$

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_{i1} - \vec{x}_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{id} - x_{jd})^2} \qquad (3)$$

diameter ของคลัสเตอร์แสดงถึงความใกล้ชิดกันระหว่างอ็อบเจกต์ภายในคลัสเตอร์ได้ โดยรูปที่ 1a จะเห็นได้ว่าคลัสเตอร์ C1 มีความหนาแน่นและความเป็นปึกแผ่นของข้อมูลสูงกว่าคลัสเตอร์ C2 และเมื่อลากเส้นเชื่อมระหว่างอ็อบเจกต์ใดๆ ไปยังทุกอ็อบเจกต์ภายในคลัสเตอร์ (ดังรูปที่ 1b) และนำความยาวกำลังสองของเส้นเชื่อมเหล่านั้นมารวมกันแล้วหารด้วยจำนวนเส้นเชื่อมทั้งหมด (12 เท่ากันทั้งสองคลัส-เตอร์) และนำมาถอดรากที่สองจะพบว่าค่าที่ได้จากคลัสเตอร์ C1 น้อยกว่าค่าที่ได้จากคลัสเตอร์ C2 แสดงให้เห็นว่าอ็อบเจกต์ของคลัสเตอร์ C1 มีความเป็นปึกแผ่นสูงกว่าอ็อบเจกต์ของคลัสเตอร์ C2



รูปที่ 1 ตัวอย่างของสองคลัสเตอร์ที่มีความหนาแน่นต่างกัน

Compactness (Θ) คือค่าความเป็นปึกแผ่นของข้อมูลแบบประมาณโดยใช้การอ่านข้อมูลเพียงรอบเดียว โดยค่า Compactness เป็นส่วนกลับของ discordancy (Γ)

หรือความไม่ประสานกันภายในกลุ่มข้อมูล โดยค่า Compactness สามารถคำนวณได้ดังสมการที่ (4)

$$\Theta = \frac{1}{\Gamma} \quad (4)$$

สำหรับค่า discordancy สามารถคำนวณได้สองแบบคือ weighted discordancy และ non-weighted discordancy

Weighted discordancy ($\vec{\Gamma}$) คือค่าความไม่ประสานกันภายในกลุ่มข้อมูลแบบให้น้ำหนัก สำหรับการคำนวณค่า $\vec{\Gamma}$ คำนวณได้ดังดังสมการที่ (5)

$$\vec{\Gamma} = \frac{\sum_{i=1}^{N}[(n'_i - 1).d(\vec{\mu}_{i-1}, \vec{x}_i)]}{\sum_{i=1}^{N}(n'_i - 1)} \quad (5)$$

โดยให้ $\vec{\mu}_0 = \vec{x}_i$, n'_i เป็นจำนวนอ็อปเจ็กต์อ้างอิงของตัวแทน และ $n'_1 = 2$

Non-weighted discordancy ($\tilde{\Gamma}$) คือค่าความไม่ประสานกันภายในกลุ่มข้อมูลแบบไม่ให้น้ำหนัก โดยค่า $\tilde{\Gamma}$ สามารถคำนวณได้ดังสมการที่ (6)

$$\tilde{\Gamma} = \frac{\sum_{i=1}^{N} d(\mu_{i-1}, x_i)}{(N-1)} \quad (6)$$

ตารางที่ 1 แสดงการคำนวณค่า $\vec{\Gamma}$, $\tilde{\Gamma}$ และ Centroid ($\vec{\mu}$) ของอ็อบเจกต์ $\vec{x}_i$

i	$\vec{\Gamma}$	$\tilde{\Gamma}$	$\vec{\mu}$
1	$0.d(\vec{\mu}_0, \vec{x}_1)$	$d(\vec{\mu}_0, \vec{x}_1)$	$\frac{\vec{x}_1}{1}$
2	$\frac{0.d(\vec{\mu}_0, \vec{x}_1) + 1.d(\vec{\mu}_1, \vec{x}_2)}{1}$	$\frac{d(\vec{\mu}_0, \vec{x}_1) + d(\vec{\mu}_1, \vec{x}_2)}{1}$	$\frac{\vec{x}_1 + \vec{x}_2}{2}$
3	$\frac{\left\{\begin{matrix}0.d(\vec{\mu}_0, \vec{x}1) + 1.d(\vec{\mu}_1, \vec{x}_2)\\ + 2.d(\vec{\mu}_2, \vec{x}_3)\end{matrix}\right\}}{1+2}$	$\frac{\left\{\begin{matrix}d(\vec{\mu}_0, \vec{x}1) + d(\vec{\mu}_1, \vec{x}_2)\\ + d(\vec{\mu}_2, \vec{x}_3)\end{matrix}\right\}}{2}$	$\frac{\vec{x}_1 + \vec{x}_2 + \vec{x}_3}{2}$
⋮	⋮	⋮	⋮
N	$\vec{\Gamma} = \frac{\sum_{i=1}^{N}[(n'_i - 1).d(\vec{\mu}_{i-1}, \vec{x}_i)]}{\sum_{i=1}^{N}(n'_i - 1)}$	$\tilde{\Gamma} = \frac{\sum_{i=1}^{N} d(\mu_{i-1}, x_i)}{(N-1)}$	$\vec{\mu} = \frac{\sum_{i=1}^{N} \vec{x}_i}{N}$

สมมุติให้ข้อมูลขนาด N อ็อบเจกต์ $x_1, x_2, ..., x_N$ ซึ่งถูกแบ่งออกเป็นคลัสเตอร์หรือกลุ่มย่อยๆ g กลุ่ม โดยแต่ละกลุ่มประกอบด้วยอ็อบเจกต์จำนวน $n_1, n_2, ..., n_g$ (จำนวนของอ็อบเจกต์แสดงถึงความหนาแน่นของข้อมูลภายในกลุ่ม) และความเป็นปึกแผ่นของข้อมูลภายในกลุ่ม (Compactness) มีค่าเท่ากับ $\Theta_1, \Theta_2, ..., \Theta_g$ เมื่อต้องการสร้างชุดข้อมูลสุ่มขนาด M อ็อบเจกต์ โดยที่ความน่าจะเป็นที่อ็อบเจกต์ x_j ใดๆ ภายในกลุ่มที่ i จะถูกเลือกจะต้อง

(i)ผกผันกับค่าความหนาแน่นของข้อมูลภายในกลุ่ม

$$P(x_j) \alpha \frac{1}{n_i} \quad (7)$$

(ii) แปรผันตรงกับค่าความเป็นปึกแผ่นของข้อมูลภายในกลุ่ม

$$P(x_j) \alpha \Theta_g \text{ หรือ } P(x_j) \alpha \frac{1}{\Gamma_i} \quad (8)$$

ดังนั้นเพื่อให้สอดคล้องกับ (i) และ (ii) จึงให้ความน่าจะเป็นที่อ็อบเจกต์ x_j ใดๆ ภายในกลุ่มที่ i จะถูกเลือกมีค่าเท่ากับ

$$P(x_j) = \frac{\alpha}{n_i^{\,e}.\Gamma_i^{\,e}} \quad (9)$$

โดยให้ α, e, และ δ เป็นค่าคงที่ใดๆ ซึ่งค่า α ที่เหมาะสมสำหรับการสุ่มที่ต้องการชุดข้อมูลสุ่มขนาด M อ็อบเจกต์ สามารถพิจารณาได้ดังดังสมการที่ (10)

$$\alpha = \frac{M}{\sum_{i=1}^{g}[n_i^{1-e}.\Gamma_i^{-\delta}]} \quad (10)$$

และเมื่อแทนค่า α ในสมการที่ (3)-(4) จะได้ความน่าจะเป็นที่อ็อบเจกต์ x_j ใดๆ ภายในกลุ่มที่ i จะถูกเลือกสำหรับชุดข้อมูลสุ่มขนาด M มีค่าเท่ากับ

$$P(x_j) = \frac{M}{n_i^{\,e}.\Gamma_i^{\,\delta}.\sum_{i=1}^{g}[n_i^{1-e}.\Gamma_i^{-\delta}]} \quad (11)$$

จากสมการข้างต้นจะเห็นได้ว่าในบางกรณีส่วนหารสามารถมีค่าเป็นศูนย์ได้ เช่น n_i มีค่าเท่ากับ 1 ซึ่งทำให้ค่า Γ_1 มีค่าเป็นศูนย์ ดังนั้นเพื่อเป็นการเลี่ยงปัญหาการหารด้วยค่าศูนย์(devide by zero) จึงปรับปรุงฟังก์ชัน $d(\vec{x}_i, \vec{x}_j)$ ที่ใช้สำหรับอัลกอริทึม DBSPACE โดยการบวก 1 เพิ่มเข้าไปกับค่าของระยะห่างที่ได้จากการคำนวณปกติ ซึ่งจะมีผลทำให้ค่า Γ ที่ได้มีค่าเพิ่มขึ้นจากเดิมประมาณ 1 มีผลทำให้ Γ^{δ} เป็นฟังก์ชันเพิ่มเสมอ ในรูปที่ 2 แสดงอัลกอริทึม DBSPACE โดยที่ $n[i]$ แสดงถึงจำนวนข้อมูลภายในกลุ่มที่ i , $lsx\,[i]$ เป็นผลรวมเชิงเวกเตอร์ของทุกอ็อบเจกต์ภายในกลุ่มที่ i , $lsd\,[i]$ คือผลรวมของระยะห่างระหว่างอ็อบเจกต์ภายในกลุ่มที่ i

$\alpha_D = 0$
FOR each input point *x* DO
i = hash($\vec{x}$)
IF *n*[*i*] <> 0 THEN
$\alpha_D = \alpha_D - (n[i]^{1-e}.discordancy(lsd[i],n[i]))^{-\delta}$
n[*i*] = *n*[*i*] + 1
IF *n*[*i*] = 1 THEN $\vec{\mu} = \vec{x}$
IF $\Gamma = \overline{\Gamma}$ THEN *lsd*[*i*] = *lsd*[*i*]+(*n*[*i*]-1).*d*($\vec{\mu}, \vec{x}$)
ELSE *lsd*[*i*] = *lsd*[*i*]+*d*($\vec{\mu}, \vec{x}$)
$ls\vec{x} = ls\vec{x} + \vec{x}$
$\vec{\mu} = ls\vec{x}/n[i]$
Γ = *discordancy*(*lsd*[*i*],*n*[*i*])
$\alpha_D = \alpha_D + (n[i]^{1-e}.\Gamma^{-\delta})$
WITH prob. P=min $\{M/[\alpha_D.n[i]^e.\Gamma^{\delta}].1\}$ DO
IF the output buffer is full THEN *reduce*()
// start drawing when group contains more than 1 object
IF *n*[*i*] > 1 THEN add < *P*, $\vec{X}$ > to the output buffer
reduce()
FOR each output buffer entry < P_i, $\vec{X}_i$ > DO output <1/P_i, $\vec{X}_i$ >

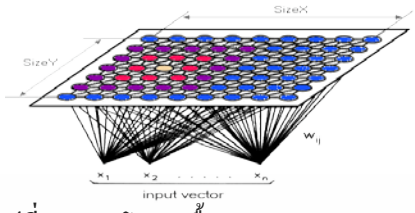
reduce() IS
FOR each output buffer entry <P_i, $\vec{X}_i$ > DO
Let P_i'=min $\{M/[\alpha_D.n[i]^e.\Gamma^{\delta}].1\}$
WITH prob. P_i'/P_i replace this entry with < P_i', $\vec{X}_i$ >
OTHERWISE remove this entry

discordancy(*lsd*, *n*) IS
IF *n*<= 1 THEN *n*=2
IF $\Gamma = \overline{\Gamma}$ THEN
disc = *lsd*/((*n**(*n*-1))/2)
ELSE
disc = *lsd*/(*n*-1)
RETURN *disc*

อัลกอริทึม DBSPACE จะทำการแบ่งกลุ่มข้อมูลและสุ่มเลือกข้อมูลไปพร้อมๆ กันแบบคู่ขนานภายในการอ่านข้อมูลเพียงหนึ่งรอบ และข้อมูลที่ถูกเลือกจะถูกเก็บไว้ในบัฟเฟอร์(output buffer) โดยในบัฟเฟอร์ประกอบด้วย $< P_i, \vec{x}_i >$ ซึ่งแสดงถึงการที่ $\vec{x}_i$ ถูกเลือกด้วยค่าความน่าจะเป็นเท่ากับ P_i และมีการรับข้อมูลเข้ามาเรื่อยๆ $\vec{x}_i$ จะมีค่าความน่าจะเป็นที่จะถูกเลือกเท่ากับ P_i' ดังนั้นจึงต้องมีการปรับปรุงค่าในบัฟเฟอร์อยู่เสมอเพื่อให้ชุดข้อมูลสุ่มที่ได้เป็นปัจจุบัน โดยการเก็บ $< P_i', \vec{x}_i >$ แทน $< P_i, \vec{x}_i >$ ด้วยความน่าจะเป็น P_i'/P_i (หรือทำการลบอ็อบเจกต์นั้นออกจากบัฟเฟอร์)

2.2 Self –Oraganizing Map

Self-Organzing Map(SOM) เป็นนิวรอนเน็ตเวิร์กแบบไม่มีผู้สอน [7] นำเสนอโดยศาสตราจารย์โคโฮเนน (Kohonen) ในปี ค.ศ. 1982 ประกอบด้วยเซลล์ 2 ชั้น ดังรูปที่ 3 ชั้นแรกคือชั้นของอินพุต (Input Layer) ประกอบด้วยเซตของอินพุตเวกเตอร์ *x(t)* (1 x n มิติ) โดยที่ i คืออินเด็กซ์ของอินพุท ชั้นที่สองคือชั้นของเอาท์พุตประกอบไปด้วยโหนดของนิวรอนที่เรียงตัวอยู่ในรูปแบบของแผนภาพ 2 มิติ ในแต่ละโหนด i จะเป็นค่าเวคเตอร์น้ำหนักแทนด้วย $w_i(t)$ นั้นคือ $w_i(t) \in \Re^n$ โดยที่ $\Re^n$



รูปที่ 3 แสดงโมเดลพื้นฐานของ SOM

กระบวนการเรียนรู้ของ SOM เกิดขึ้นจากการปรับตัวของเวคเตอร์น้ำหนักที่มีต่ออินพุตเวคเตอร์โดยเริ่มแรกจะกำหนดน้ำหนักเริ่มต้นขนาดเล็กให้กับทุกโหนด จากนั้นจะเริ่มต้นกระบวนการเรียนรู้ดังนี้

1. เลือกอินพุตเวคเตอร์จากอินพุตโดเมน
2. เปรียบเทียบอินพุตเวคเตอร์ *x(t)* กับโหนด *w(t)* ทุกโหนดเพื่อหาโหนดชนะจากโหนดทั้งหมด
3. ปรับเวคเตอร์น้ำหนักของโหนดชนะ เพื่อให้โหนดชนะเข้าใกล้อินพุตมากขึ้น
4. ปรับเวคเตอร์น้ำหนักของโหนดใกล้เคียง เพื่อให้อินพุตเวคเตอร์ถัดไปที่มีค่าใกล้เคียงกับโหนดชนะ กระบวนการเหล่านี้จะถูกทำซ้ำไปเรื่อย ๆ จนกว่าจะสอดคล้องตามเงื่อนไข

กระบวนการเรียนรู้ข้างต้นมีการคำนวณที่สำคัญอยู่ 2 ส่วนคือ ส่วนแรกคือการคำนวณเพื่อหาโหนดชนะ c คือโหนดที่มีระยะห่างระหว่างอินพุตเวคเตอร์กับเวคเตอร์น้ำหนักน้อยที่สุด โดยใช้ฟังก์ชันวัดระยะทางแบบยุคลิด (Euclidean distance) โหนดชนะหาดังสมการที่ 12

$$c : w_c t = \min \| x(t) - w_i(t) \| \quad (12)$$

ส่วนที่สองคือการปรับเวคเตอร์น้ำหนักเพื่อให้เข้าใกล้อินพุตมากขึ้น การปรับค่าน้ำหนักแสดงดังสมการที่ 13

$$w_i(t+1) = w_i(t) + \alpha(t) \times h_{ci}(t) \times [x(t) - w_i(t)] \quad (13)$$

เมื่อ t คือรอบปัจจุบันของการเรียนรู้, x(t)คืออินพุตเวคเตอร์ปัจจุบัน, $w_i(t)$คือเวคเตอร์น้ำหนัก, $\alpha(t)$ คืออัตราการเรียนรู้ โดยที่อัตราการเรียนรู้ $\alpha(t)$ จะขึ้นอยู่กับจำนวนรอบซึ่งแสดงเป็นสมการเชิงเส้นได้ดังสมการที่ 14

$$\alpha(t) = \alpha(0) \times \frac{T-t}{T} \qquad (14)$$

เมื่อ T คือจำนวนรอบทั้งหมด, t คือจำนวนรอบปัจจุบัน h_{ci} คือฟังก์ชั่นที่ใช้ในการกำหนดน้ำหนักในการปรับค่าโหนดใกล้เคียงโดยทั่วไปแล้ว จะใช้ฟังก์ชันเกาส์เซียน ซึ่งสามารถเขียนได้ดังสมการที่ 15

$$h_{ci}(t) = \exp(-\frac{\| r_c - r_i \|^2}{2\sigma^2(t)}) \qquad (15)$$

โดยปกติรัศมีของโหนดใกล้เคียงจะค่อย ๆ ลดลงตามจำนวนรอบในการเรียนรู้ t ดังสมการที่ 16

$$\sigma(t+1) = 1 + (\sigma(t) - 1) \times \frac{T-t}{T} \qquad (16)$$

2.3 ขั้นตอนของวิธีที่นำเสนอ

ข้อมูลอินพุทที่ใช้เรียนรู้ในแผนภาพมี 2 ส่วนโดยส่วนแรกเป็นข้อมูลเอาท์พุทที่ได้จาก DBSPACE และส่วนที่ 2 เป็นข้อมูลอินพุททั้งหมดโดยข้อมูลส่วนแรกเรียนรู้ประมาณ 70 เปอร์เซ็นต์ของจำนวนรอบการเรียนรู้สูงสุด และข้อมูลส่วนที่ 2 จะเรียนรู้อีกประมาณ 30 เปอร์เซ็นต์ซึ่งขั้นตอนการทำงานแสดงได้ดังนี้

กำหนดให้ชุดข้อมูลฝึกฝนอินพุทเป็น $x_1, x_2, \ldots, x_K$ เมื่อ $x_k = [x_{k1}, \ldots, x_{kR}]^T$, k=1,...,K, และ K เป็นจำนวนข้อมูลทั้งหมดในชุดฝึกฝน L_{max} เป็นจำนวนรอบสูงสุดในการเรียนรู้ที่กำหนด

ขั้นตอนที่ 1.นำข้อมูลอินพุท $x_1, x_2, \ldots, x_K$ ผ่านกรรมวิธีการลดข้อมูลโดย DBSPACE ได้ข้อมูลสุ่มชุดหนึ่ง $y_1, y_2, .., y_J$ ซึ่งเป็นเอาท์พุทของ DBSPACE

ขั้นตอนที่2.สุ่มค่าน้ำหนัก w_i i=1,2,...M

ขั้นตอนที่3.รับข้อมูลชุดฝึกสอนจากเอาท์พุทของ DBSPACE เข้าโครงข่ายคำนวณระยะทางระหว่าง$[y_j - w_i]$

ขั้นตอนที่4.หาค่านิวรอนที่ชนะจากนั้นปรับค่าเวกเตอร์น้ำหนัก ของประสาทเทียม

ขั้นตอนที่5.ถ้า j < J ให้ j=j+1 ทำการคำนวณซ้ำจากขั้นตอนที่ 3

ขั้นตอนที่6.ถ้าจำนวนรอบ m < (L_{max} –L_{max}*0.3) (0.3 คือ 30 เปอร์เซ็นต์) กำหนด j=1,m=m+1 เริ่มรอบฝึกฝนใหม่จากขั้นตอนที่ 3

ขั้นตอนที่7.รับข้อมูลชุดฝึกสอน $x_1, x_2, \ldots, x_K$ เข้าโครงข่ายคำนวณระยะทางระหว่าง$[x_k - w_i]$

ขั้นตอนที่8.หาค่าประสาทเทียมที่ชนะปรับค่าเวกเตอร์น้ำหนักของประสาทเทียม

ขั้นตอนที่9.ถ้า k< K ให้k=k+1 ทำการคำนวณซ้ำจากขั้นตอนที่ 7

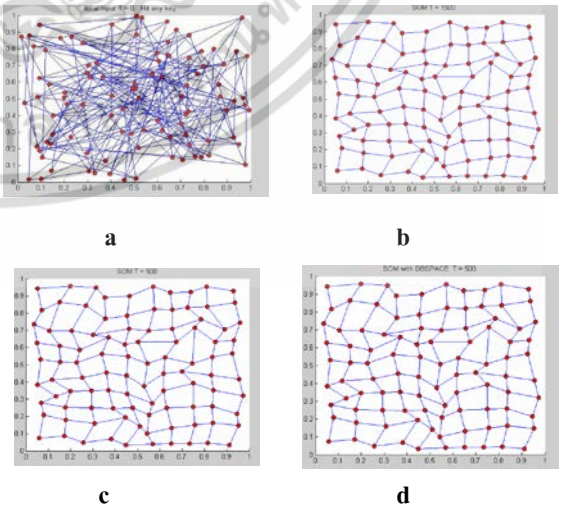
ขั้นตอนที่10.ถ้าจำนวนรอบ m < L_{max} กำหนด k=1,m=m+1 เริ่มรอบฝึกฝนใหม่จากขั้นตอนที่ 7

3.การทดลองและผลการทดลอง

ทำการทดลองโดยเครื่องคอมพิวเตอร์มีหน่วยประมวลผลเป็น Intel Core2Duo 2.0GHz Memory 2 GB ระบบปฏิบัติการ Windows XP

3.1 การทดลองที่ 1

ชุดข้อมูลที่ใช้มีค่าระหว่าง0-1โดยการสุ่ม เป็นเวคเตอร์ 2 มิติจำนวน 1000 เวคเตอร์ นำมาทำการเรียนรู้ด้วย SOM แบบเดิมและแบบที่นำเสนอขนาดโมเดล 10×10 โดยกำหนดอัตราการเรียนรู้เริ่มต้น α =0.5 จะเห็นได้ว่าเมื่อทำการเรียนรู้ในแผนภาพ SOM แบบเดิมจำนวน 1500 รอบรูปที่ 4b และ 500 รอบรูปที่ 4c แผนภาพของโหนดที่แสดงไม่แตกต่างกันแสดงว่าการเรียนรู้เพียง 500 รอบก็เพียงพอในการทดสอบและเมื่อนำข้อมูลมาเรียนรู้ผ่าน SOM ที่นำเสนอโดยเรียนรู้ 500 รอบ แผนภาพที่ได้เหมือนกันกับที่เรียนรู้โดย SOM แบบเดิมดังรูปที่ 4d ส่วนเวลาที่ใช้ในการเรียนรู้ลดลงแสดงดังตารางที่ 2



a b c d

รูปที่ 4 แสดงแผนภาพของ SOM

ตารางที่ 2 ผลการทดลองในการเรียนรู้ข้อมูลสุ่ม

วิธีการ	จำนวนรอบ	เวลา(วินาที)
SOM	500	150.89
Our Method	500	67.62

3.2 การทดลองที่ 2

ชุดข้อมูล ecoli[8] ซึ่งมีข้อมูลทั้งหมด 336 ชุด 8 แอทริบิวต์โดยนำมาทดลองจำนวน 5 กลุ่ม จำนวน 327 ข้อมูล ทดลองโดยผ่าน SOM แบบเดิมและแบบที่นำเสนอ ขนาดโมเดล 4×4 โดยกำหนดอัตราการเรียนรู้เริ่มต้น α =0.5 และวัดประสิทธิภาพโดยการวัดค่าแบบเอนโทรพี (entropy) คือ การวัดโดยอาศัยความน่าจะเป็นของข้อมูลที่อยู่ในกลุ่ม สามารถหาได้จาก

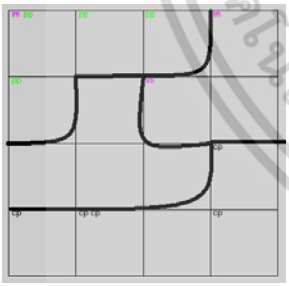
$$E_j = -\sum_i p_{ij} \log(p_{ij}) \qquad (16)$$

โดยที่ค่าเอนโทรพีรวมของทุกกลุ่มสามารถหาได้จาก

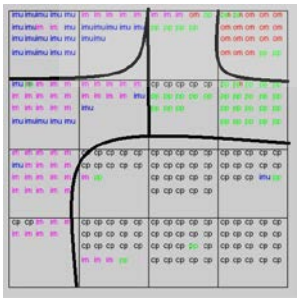
$$E_{cs} = \sum_{j=1}^{m} \frac{n_j \times E_j}{n} \qquad (17)$$

โดยที่ p คือความน่าจะเป็นที่สมาชิกของโหนดขึ้นอยู่กันกลุ่ม j, n_j คือ จำนวนข้อมูลทั้งหมดในโหนด j, n คือ จำนวนเอกสารทั้งหมด

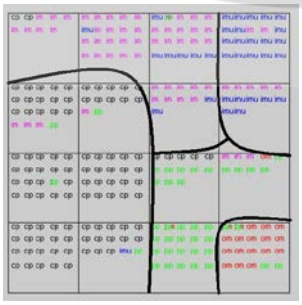
ได้ผลการทดลองดังตารางที่ 3 และแผนภาพดังรูปที่ 5



a) แสดงแผนภาพหลังจากเรียนรู้ด้วยอินพุทจากDBSPACE

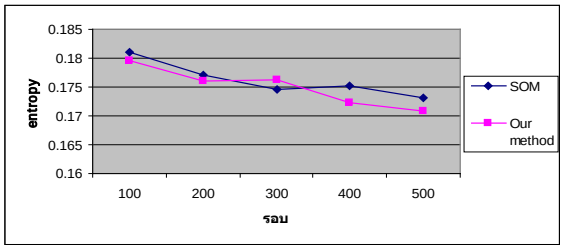


b) Som



c) Our method

รูปที่ 5 แสดงแผนภาพที่ได้จากข้อมูลชุด ecoli



รูปที่ 6 กราฟของจำนวนรอบการทำงานและค่าเอนโทรพี

ตารางที่ 3 ผลการทดลองในการเรียนรู้ชุดข้อมูล ecoli

วิธีการ	จำนวนรอบ	เวลา(วินาที)	ค่าเอนโทรพี
SOM	1500	42.062	0.17149
Our Method	1500	16.156	0.16702

4.สรุป

การทดลองแสดงว่าวิธีที่นำเสนอใช้เวลาในการเรียนรู้ลดลงเมื่อเปรียบเทียบกับ SOM แบบเดิมเนื่องจากการประยุกต์ใช้ DBSPACE นั้นช่วยทำให้จำนวนข้อมูลที่ใช้เรียนรู้ลดลงและข้อมูลที่ได้เป็นตัวแทนกลุ่มของข้อมูลทั้งหมด เมื่อนำข้อมูลนั้นไปฝึกในโครงข่ายก่อนใช้ข้อมูลเดิม จึงส่งผลให้การจัดเรียงตัวของแผนภาพทำได้เร็วขึ้นและใช้เวลาในการเรียนรู้ลดลงมากกว่า 50 เปอร์เซ็นต์แต่ประสิทธิภาพในการจัดกลุ่มข้อมูลยังเหมือนเดิม

5.เอกสารอ้างอิง

[1] J.Kangas. “Time dependent self-organizing maps for Speech recognition.” ICANN-91,Espoo, June 24-28,1991

[2] Kaski S.,Honkela T.,Lagus K.,and Kohonent T. “WEBSOM Self-organizing maps of document collections”Neurocomputing,Volume21,1998,pp.101-117.

[3] Samuel Kaski, Jari Kangast, Teuvo Kohonen. “Bibiography of Self-Organizing Map(SOM) Paper: 1981-1977” Neural Computing Survey I, 1998.

[4] Mu-Chunn Su,Hsiao-Te Chang. “Fast Self-Organizing Feature Map Algorithm.” IEEE Transaction on neural Networks vol.11,no.3,May 2000.

[5] Kaski,S. “Fast winner search for SOM-based monitoring and retrieval of high-dimensional data” Proceedings of ICANN99 Ninth International Conference on Artificial Neural Networks, London, vol 2, pp940-945

[6] ธรรมศักดิ์ เธียรนิเวศน์ “การลดขนาดข้อมูลด้วยน้ำหนักความหนาแน่นเพื่อการจัดกลุ่มของมูลขนาดใหญ่” วิทยานิพนธ์ปริญญามหาบัณฑิต มหาวิทยาลัยสุรนารี 2548

[7] Helge Ritter, Thomas Martinetz and Klaus Schulten. Neural computation and organizing maps:an introduction Massachusetts:Addison-wesley. 1992

[8] http://archive.ics.uci.edu/ml/ datasets/ecoli