

שם הקורס: מבוא לנוירן-חישוביות

מס' קורס : 7079010  
מרצה : פרופ' לרי מנביץ  
העורך : משה חנוקוגלו, רעות דביר, אהוד פלקסין  
תאריך : תשע"ח סמ' ב

2	.....	Binary and bipolar representation
3	.....	The McCulloch-Pitts Neuron
3	.....	מבנה הנוירון
3	.....	משפט
5	.....	The Perceptron
5	.....	אלגוריתם
5	.....	משפט ההתכנסות של הפרספטרון
5	.....	הוכחה
7	.....	Feed Forward Networks and Back-Propagation
7	.....	ארכיטקטורה
7	.....	אלגוריתם
9	.....	Discrete Hopfield Net
9	.....	אלגוריתם
9	.....	נוכיח כי האלגוריתם מסתיים
10	.....	נוכיח כי חסום מלמטה
10	.....	שימושים
11	.....	Bam
11	.....	אלגוריתם
12	.....	נוכיח כי האלגוריתם מסתיים
12	.....	שימושים
13	.....	Maxnet
13	.....	אלגוריתם
14	.....	Hamming net
14	.....	Hamming distance
14	.....	אלגוריתם
15	.....	Kohonen Network
15	.....	אלגוריתם
16	.....	Counter Propagation Network
17	.....	Sparse Distributed Memory
18	.....	TSDM
19	.....	Hebb rule

## Binary and bipolar representation

---

הערה : בסוף כל פרק כתוב איזה פרקים מומלצים לקריאה בספר בנושא.

הספר מובא הקישור הבא :

<http://www.csbdu.in/csbdu-old/pdf/Fundamentals%20Of%20Neural%20Networks.pdf>

וקטור בקורס זה יהיה במבנה הבא :

$$(x_1, x_2, \dots, x_n)$$

אך ישנן כמה שיטות לערכים של  $x_i$ , נציג את השתיים העיקריות והן bipolar, binary.

binary  $x_i \in \{0,1\}$  אולם ב bipolar  $x_i \in \{-1,1\}$  (יש שמוסיפים ל bipolar גם את המספר 0).

ל bipolar ישנם יתרונות על פני binary, למשל ישנן בעיות שלא פתירות בייצוג binary אבל בייצוג bipolar הן פתירות.

כמו כן, בייצוג bipolar אפשר להבדיל בין קורדינטה לא ידוע שנשמנה ע"י 0 לבין קורדינטה שגויה שנשמנה ע"י 1.

עמודים בספר : 48

## The McCulloch-Pitts Neuron

כל נוירון בנוי בצורה הבאה, ישנן  $n$  חיוביות ובנוסף יש לו  $m$  כניסות שליליות כאשר לכל כניסה יש משקל, לכל הכניסות החיוביות משקל חיובי וזהו וכן לכל הכניסות השליליות יש אותו משקל שלילי. וישנו מספר שנחשב סף (threshold) מספר זה משמש כאינדקציה לנוירון האם להוציא 0 או 1.

### מבנה הנוירון

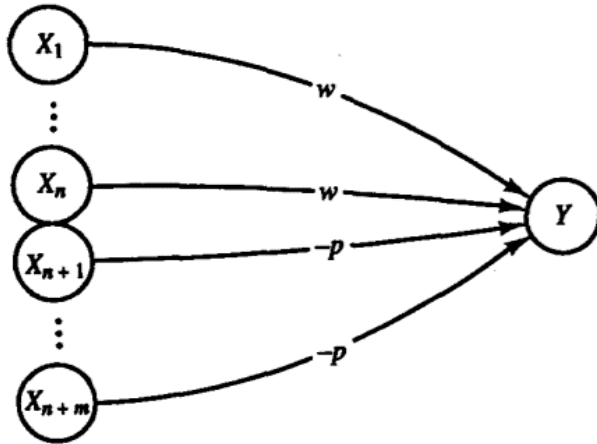


Figure 1.13 Architecture of a McCulloch-Pitts neuron Y.

ע"מ שהנוירון ידע אם להוציא 0 או 1 הוא בודק מה המצב של הקלטים שלו, כלומר הוא מחשב מה הערך של כל כניסה (לכניסה  $i$ ) אם ערך הקורדינטה הינו ("דלוק") 1 אזי ערך הכניסה הזו שווה למשקל של הכניסה הזו, אך אם ערך הקורדינטה הוא 0 (או -1 ב bipolar) אזי ערך כניסה זו שווה ל 0 (או למינוס המשקל ב bipolar).

אם סכום כל הכניסות החיוביות שלו גדול מהסף אזי הנוירון מוציא 1 אך אם אפילו כניסה שלילית אחת שלו דלוקה אז הוא מוציא 0 כמו כן אם הסכום של הכניסות החיוביות שלו לא עבר את הסף אזי הוא מוציא 0.

ובצורה פורמלית Y יירה אם הוא יקבל  $k$  או יותר קלטים מגרים ובכלל לא קלטים מעכבים כאשר:

$$kw \geq \theta > (k - 1)w$$

ואין שום כניסה שלילית דלוקה.

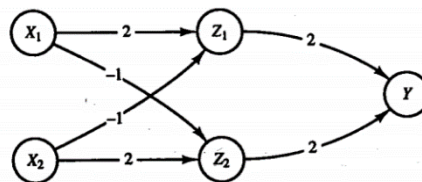
רשת נוירונים הכוונה היא שישנם כמו נוירונים שמחוברים אחד לשני (הפלט של נוירון אחד הוא הקלט של נוירון אחר)

על ידי רשת נוירונים נוכל לבנות כל פונקציה לוגית. שהרי ניתן ליצור על ידי נוירון בודד את הפונקציות AND, OR, NOT ומכיוון שכל פונקציה לוגית ניתן לכתוב בצורת DNF כלומר שתהיה שכבת נוירונים שיבצעו NOT שכבה שניה תבצע AND ושכבה שלישית תבצע OR

ולכן ניתן לממש XOR שמשוואתו בצורת DNF היא:

$$x_1 \text{ XOR } x_2 \leftrightarrow (x_1 \text{ AND NOT } x_2) \text{ OR } (x_2 \text{ AND NOT } x_1).$$

באופן הבא:



כאשר הסף של כל נוירון הוא 2.

### משפט

כל פונקציה לוגית (בוליאנית) עם כל מספר של משתנים ניתן לייצג על ידי רשת McCulloch-Pitts

## הוכחה

שימוש ב DNF והצגת NOT, AND, OR

כל כניסה שיש לה בהצגת DNF שלילה תכנס בתחילה לפרספטרון של NOT, כל זוג תוצאות (אילו שנכנסו לפרספטרון של NOT ואילו שלא) יכנסו יחד לפרספטרון של AND בהתאם להצגת DNF, וכל התוצאות האלו יכנסו בזוגות גם כן לפרספטרון OR בהתאם להצגת DNF, יחושב בסף ויחליט אם עובר או לא. מכאן קיבלנו שניתן ליצור כל פונקציה בולינאית על ידי 3 שכבות, כלומר שכבת NOT, שכבת AND, שכבת OR.

עמודים בספר : פסקה 1.6

פרספטרוני בנוי משלשה של : סנסור, אסוציאטור, רספונס. הוא משתמש בפונקציה בינארית לסנסור ולאסוציאטור, ובפונקציית אקטיבציה  $1,0,-1$  לתגובה. כאשר שכבת הסנסור מחוברת לשכבת האסוציאציה על ידי משקלים קבועים.

ע"מ להפוך את הסף להיות תמיד 0 נוסף עוד מימד לכניסה של הפרספטרוני שהמשקל שלו יהיה כמו מינוס הסף והכניסה שלו תמיד תהיה "דלוקה". לכניסה זו נקרא bias.

הבדלים בין פרספטרוני לנוירון McCulloch-Pitts :

1. המשקלים והסף לא חייבים להיות זהים
2. המשקולות יכולות להיות חיוביות או שליליות
3. אין מדכא מוחלט, כלומר יכולה להופיע משקולת שלילית ועדיין נקבל תוצאה
4. למרות שהנוירונים עדיין בשני מצבים, פונקציית התוצאה נעה בין  $[-1,1]$  ולא  $[0,1]$
5. הכי חשוב- יש כלל למידה.

## אלגוריתם

1. התחל את המשקולות ( $w$ ) להיות 0 ואת bias ( $b$ ) להיות מינוס הסף.
2. כל עוד לא הגענו למצב עצירה בצע את השלבים הבאים
  - 2.1. לכל אחד מהוקטורים של האימון שמוצגים כזוג (הוקטור ( $s$ ) והערך שמצפים שהנוירון יוצא לוקטור ( $t$ )) בצע את השלבים הבאים
    - 2.1.1. הכנס את הוקטור לקלטים של הפרספטרוני
    - 2.1.2. חשב את פונקציית האקטיבציה של הפרספטרוני

$$y_{in_j} = b_j + \sum_i x_i w_{ij}$$

$$y_j = \begin{cases} 1 & \text{if } y_{in_j} > \theta \\ 0 & \text{if } -\theta \leq y_{in_j} \leq \theta \\ -1 & \text{if } y_{in_j} < -\theta \end{cases}$$

2.1.3. בדוק אם  $t \neq y_j$  אזי

$$b(new) = b(old) + t \quad 2.1.3.1$$

$$w_i(new) = w_i(old) + t * x_i \quad 2.1.3.2$$

2.2. אם לא התבצע שום שינוי (כניסה לשלבים 2.1.3.1 ו 2.1.3.2) אזי תעצור.

## משפט ההתכנסות של הפרספטרוני

אם קיים וקטור משקולות  $w^*$  כך ש  $f(x(p) * w^*) = t(p)$  לכל  $p$  אזי עבור כל וקטור משקולות התחלתיים שהוא, כלל הלמידה של הפרספטרוני יתכנס לוקטור משקולות אשר יביא את התגובות הנכונות עבור כל אחד מהדוגמאות הנלמדות, ויעשה זאת בכמות סופית של צעדים.

## הוכחה

בלי הגבלת הכלליות נניח כמה דברים :

1. כל הדוגמאות הן חיוביות (כל דוגמא שלילית נהפוך לדוגמא חיובית, על ידי הכפלת הדוגמא במינוס).
2. נניח שהסף הוא 0 (נוכל לתקן אותו ל-0 על ידי הוספת ביט נוסף לכל דוגמא שהוא תמיד חיובי ויאזן את הסף).
3. בלי הגבלת הכלליות נניח שכל וקטור דוגמא הוא מנורמל.

נסתכל הגדרת  $\cos$  :

$$\frac{v^* w^{n+1}}{|w^{n+1}|} \leq 1$$

נסמן  $\delta = \min(v^* x)$  הוא כל אחד מהדוגמאות.

נסתכל על המונה של האי שוויון

$$v^* w^{n+1} = v^* (w^n + x_1) = v^* w^n + v^* x_1 = v^* w^{n-1} + v^* x_2 + v^* x_1 = \dots \geq n\delta$$

נסתכל על המכנה

$$|w^{n+1}|^2 = w^{n+1} * w^{n+1} = (w^n + x)(w^n + x) = |w^n|^2 + 2w^n x + x^2$$

$$|x| = 1 \rightarrow x^2 = 1$$

$$w^n x < 0$$

מתקיים מאחר שהיינו צריכים לתקן את המשקולות בשלב ה n, לכן

$$|w^n|^2 + 2w^n x + x^2 < |w^n|^2 + 1 < \dots < n$$

לכן

$$|w^{n+1}|^2 < n$$

$$\sqrt{n}\delta = \frac{n\delta}{\sqrt{n}} < \frac{v^* w^{n+1}}{|w^{n+1}|} \leq 1$$

$$\sqrt{n} \leq \frac{1}{\delta}$$

$$n \leq \frac{1}{\delta^2}$$

קיבלנו כי n חסום במספר סופי ולכן אחרי כמות סופית של איטרציות האלגוריתם יסתיים.

כיוון שגם וקטורי הדוגמא וגם  $v^*$  מנורמלים אז מכפלה בניהם נותנת תוצאה קטנה מ 1 ולכן גם  $\delta$  קטן מ 1 דבר שגורר ש  $\frac{1}{\delta^2}$  גדול מ 1.

עמודים בספר : פיסקה 2.3

## Feed Forward Networks and Back-Propagation

### ארכיטקטורה

רשת נוירונים רב שכבתית עם שכבות נסתרות (לא הקלט ולא הפלט). כל נוירון בכל אחת מהשכבות מחובר לכל אחד מהנוירונים בשכבה הבאה. לשכבות הנסתרות ולשכבת הפלט יש bias. ה bias פועל כמשקולת שפונקציית האקטיבציה של תמיד 1. במהלך שלב החלחול אחורה (הלימוד) נשלחים ברורס לכיוון הקלט.

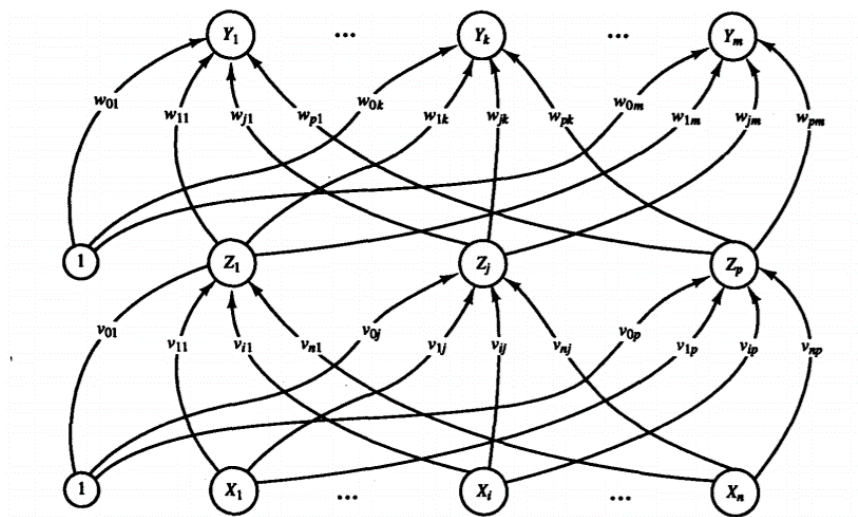


Figure 6.1 Backpropagation neural network with one hidden layer.

\*\* פונקציית האקטיבציה הינה פונקציית סיגמוייד  $f(x) = \frac{1}{1+e^{-x}}$

$$f(x) = \begin{cases} 1 & y_{in} > \sigma \\ 0 & y_{in} < \sigma \end{cases} \quad \text{פונקציה זו מאוד דומה לפונקציה}$$

### אלגוריתם

האלגוריתם מורכב משלושה חלקים:

1. Feedforward - לימוד הקלט במערכת.
2. Backpropagation - חישוב טעות הלמידה בכל אחת מהשכבות על ידי חלחול אחורה.
3. תיקון המשקולות בהתאם לחישוב הטעות.

במהלך השלב הראשון כל נוירון קלט מקבל קלט ושולח אותו לכל אחד מהנוירונים בשכבה הבאה, כל נוירון בשכבה הבאה מחשב את פונקציית האקטיבציה ושולח סיגנל לכל מהנוירונים בשכבה הבאה אחריו, וכן הלאה עד שמגיעים לשכבת הפלט. בשלב השני, בשכבת הפלט כל נוירון מחשב את פונקציית האקטיבציה שלו ומשווה אותה לערך המצופה וקובע את השגיאה,

$$\delta_k = (t_k - y_k) f'(y_{in_k})$$

$$y_k = f(y_{in_k})$$

כאשר

$t_k$  הוא הערך המצופה

$$y_{in_k} \text{ הינה ה bias שמחובר לנוירון } + \sum_{j=1}^p z_j w_{jk}$$

כל אחד מהנוירונים בכל אחת מהשכבות יבצע את החישוב הנ"ל בהתאם לשכבה שלו ויעביר את התוצאות לשכבה לפניו. לאחר שכל החישובים נעשו בכל אחת מן השכבות נעדכן את המשקולות בהתאם (וגם את bias):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

כאשר :

$$\Delta w_{jk} = \alpha \delta_k z_j$$

Momentum = עדכון המשקולות על סמך גראדינט נוכחי, גראדינטים וניתוח של מידע קודמים.

זאת על מנת שבמקרה ויש מידע שנבדק שהוא קיצוני או תקול נרצה שישפיע פחות.

$$W_{jk}(t + 1) = W_{jk}(t) + \alpha \delta_k Z_j + \mu [W_{jk}(t) - W_{jk}(t - 1)]$$

$$\mu \in (0,1)$$

דוגמאות לשימוש ברשת :

1. הפיכת טקסט לדיבור
2. הפיכת שפת סימנים לדיבור
3. (לא בטוח שדוגמא זו שייכת לנושא זה) דחיסת נתונים ו one class classification ע"י בניית רשת בעלת שלוש שכבות. כאשר בשכבה הראשונה ישנם  $x$  נירונים בשנייה ישנם  $\frac{x}{2}$  נירונים ובשלישית ישנם  $x$  נירונים עכשיו אנו נלמד את הרשת לקבל תמונה של ילד ולהוציא בדיוק את אותה התמונה על תמונה של ילד (אימון רק על דוגמאות חיוביות). אחרי אימון הרשת נוכל להזין לרשת מידע ולבחון מה יוצא, אם יוצא מה שהכנסנו אזי זו תמונה של ילד. אך אם המידע שיוצא שונה לחלוטין מהמידע שהוכנס לרשת אזי זו לא תמונה של ילד. כמו כן הראנו עכשיו שיטה לדחיסת נתונים שהרי אם הצלחנו לקחת תמונה שנכנסת ב  $x$  נירונים ולהעביר אותה דרך  $\frac{x}{2}$  נירונים אזי שהצלחנו להחזיק את התמונה ע"י פחות מ  $x$  נירונים דבר שאומר שהצלחנו לדחוס את התמונה.

[ ההבדל בין אימון one-class לבין אימון two-class הוא שבאימון one-class הרשת מתאמנת על class אחד ויודעת בסוף האימון להבחין בין class אחד לclass שני. באימון two-class מאמנים את הרשת על כל הclasses וכך היא לומדת להבחין בין הclasses השונים]

עמודים בספר : פרק 6 (בעיקר פסקה 6.1 וקטעים מפסקה 6.2)



## Discrete Hopfield Net

רשת שבה כל אחד מהנוירונים מחובר לכל נוירון. הרשת משמשת לזיכרון אסוציאטיבי, כלומר היא תזכור כמות מסוימת של תבניות של וקטורים, ובהינתן תבנית חדשה הרשת תדע לשייך אותה לתבניות המאוחסנות אצלה.

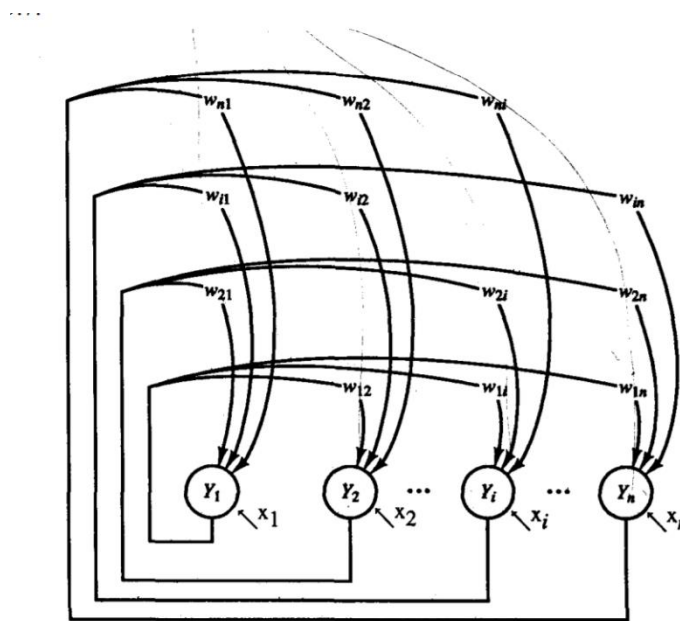
אחסון הזיכרונות נעשה על ידי בניית מטריצה ריבועית של משקולות שמחושבים באופן הבא:

נסמן את המטריצה ב  $w$ , ו  $p$  הוא וקטור שמאוחסן בזיכרון.

$$w_{ij} = \sum_p [2s_i(p) - 1][2s_j(p) - 1] \quad \forall i \neq j$$

וגם

$$w_{ii} = 0$$



### אלגוריתם

1. אתחל את המשקולות כפי שמוסבר למעלה.
2. כל עוד לא התכנסת בצע את השלבים הבאים:
3. לכל וקטור קלט  $(x)$

- 1.1 הגדר את האקטיבציה של נוירון  $y_i = x_i$
- 1.2 לכל אחד מהנוירונים בצע את הפעולות הבאות:
  - 1.2.1 חשב את הקלט:

$$y_{in_i} = x_i + \sum_j y_j w_{ji}$$

- 1.2.2 קבע את פונקציית האקטיבציה:

$$y_i = \begin{cases} 1 & \text{if } y_{in_i} > \theta_i \\ y_i & \text{if } y_{in_i} = \theta_i \\ 0 & \text{if } y_{in_i} < \theta_i \end{cases}$$

- 1.2.3 שלח לכל שאר הנוירונים את הערך של פונקציית האקטיבציה

4. תבדוק האם מתכנס

### נוכיח כי האלגוריתם מסתיים

הקדמה לחישובים:

על מנת להוכיח שהאלגוריתם מסתיים נדמה את האלגוריתם לפונקציה ונראה שהפונקציה מתכנסת תמיד ולכן גם האלגוריתם מתכנס ומסתיים.

פונקציית energy הינה פונקציה מקבוצת המצבים למספר שלם :

$$E(x_1, x_2, \dots, x_n) = -\frac{1}{2} \sum_i \sum_{j \neq i} x_i w_{ji} x_j$$

פונקציה זו חסומה על ידי קבוע מלמטה, נוכיח שהיא תמיד יורדת וכן לאחר כמות סופית של צעדים היא תתכנס לגבול שלה.

$$\begin{aligned} \exists i^* \rightarrow x_{i^*}^{old} &= -x_{i^*}^{new} \quad \forall i \neq j \quad x_i^{old} = x_j^{new} \\ \nabla E^{new} - E^{old} &= E(x^{new}) - E(x^{old}) \\ &= -\frac{1}{2} \sum_i \sum_{j \neq i} x_i^{new} w_{ji} x_j^{new} - \left( -\frac{1}{2} \sum_i \sum_{j \neq i} x_i^{old} w_{ji} x_j^{old} \right) \\ &= -\frac{1}{2} \sum_i \sum_{j \neq i} x_i^{new} w_{ji} x_j^{new} - \left( -\frac{1}{2} \sum_i \sum_{j \neq i} x_i^{old} w_{ji} x_j^{new} \right) \\ &= -\frac{1}{2} \sum_{j \neq i^*} x_{i^*}^{new} w_{ji^*} x_j^{new} + \frac{1}{2} \sum_{j \neq i^*} x_{i^*}^{old} w_{ji^*} x_j^{new} \\ &= -\frac{1}{2} (x_{i^*}^{new} - x_{i^*}^{old}) \sum_{j \neq i^*} w_{ji^*} x_j^{new} \end{aligned}$$

נדון בשני מקרים :

$$\begin{aligned} 0 &> -\frac{1}{2}(2)(+) \leftarrow 0 < \sum_{j \neq i^*} w_{ji^*} x_j^{new} \leftarrow x_{i^*}^{old} = -1 \quad \text{וגם } x_{i^*}^{new} = 1 \quad .1 \\ 0 &> -\frac{1}{2}(-2)(-) \leftarrow 0 > \sum_{j \neq i^*} w_{ji^*} x_j^{new} \leftarrow x_{i^*}^{old} = 1 \quad \text{וגם } x_{i^*}^{new} = -1 \quad .2 \end{aligned}$$

יוצא שבכל מקרה התוצאה של ה new קטנה מהתוצאה של ה old.

### נוכיח כי חסום מלמטה

$$E(\vec{x}) > -\sum_{j \neq i} |w_{ji}|$$

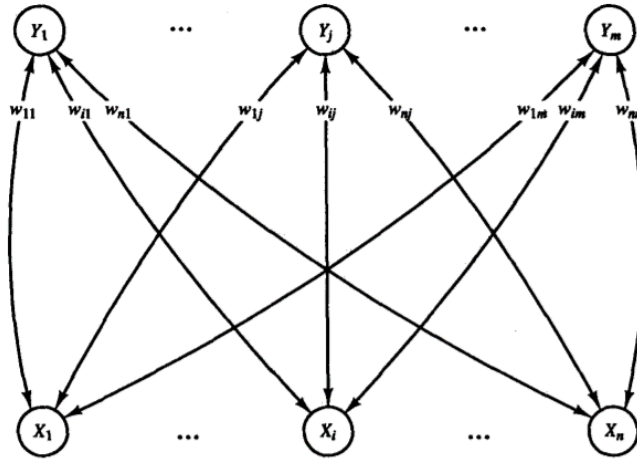
### שימושים

הרשת Hopfield משמשת לקביעת האם וקטור קלט הוא וקטור ידוע או לא ידוע. הרשת מזהה וקטור ידוע על ידי יצירת תבנית אקטיבציה על הנוירונים שהיא זהה לווקטור המאוחסן ברשת. אם וקטור הקלט הוא לא ידוע, אז וקטורי האקטיבציה במהלך האיטרציות על הרשת יתכנסו לווקטור אקטיבציה שלא מאוחסן ברשת.

לדוגמא : אם במערכת ישנם תמונה של ילד אזי אם ניתן תמונה פחות ברורה של הילד המערכת תדע לקשר את התמונה הלא ברורה של הילד.

עמודים בספר : פרק 3.4.4

רשת זו בנויה כשתי שכבות (שכבה אחת עם  $n$  נוירונים ושכבה שנייה עם  $M$  נוירונים), כך שכל נוירון בשכבה אחת מחובר לכל אחד מהנוירונים בשכבה השנייה. כל נוירון שולח אותות לכל אחד מהנוירונים בשכבה השנייה עד שכל שכבה של נוירונים מגיעה למצב שיווי משקל.



המידע שמאוחסן מגיע כזוגות של וקטורים. אחסון הזיכרונות נעשה על ידי בניית מטריצה ריבועית של משקולות שמחושבים באופן הבא:

$$w_{ij} = \sum_p [2s_i(p) - 1][2t_j(p) - 1] \quad \forall i \neq j$$

כאשר  $t$  מייצג וקטור אחד מהזוג  $s$  ו  $s$  את השני.

### אלגוריתם

נסמן את השכבות ב  $Y, X$ .

1. אתחל את מטריצת המשקולות
2. עבור כל אחד מהקלטי ניסוי עשה את השלבים הבאים:
  - 2.1 אתחל את נוירוני שכבת  $X$  כוקטור הקלט עם הגודל המתאים
  - 2.2 אתחל את נוירוני שכבת  $Y$  כוקטור הקלט עם הגודל המתאים  
אם אחד מהאתחולים (2.2 2.1) לא ידוע אתחל באפסים.
  - 2.3 כל עוד הפונקציה לא התכנסה בצע את השלבים הבאים:
    - 2.3.1 עדכן את האקטיבציה של שכבת  $Y$  באופן הבא:  
חישוב הקלט:

$$y_{in_j} = \sum_i w_{ij} x_i$$

חישוב האקטיבציה:

$$y_j = f(y_{in_j})$$

שלח את הסיגנל לשכבת  $X$ .

- 2.3.2 עדכן את האקטיבציה של שכבת  $X$  באופן הבא:  
חישוב הקלט:

$$x_{in_i} = \sum_j w_{ij} y_j$$

חישוב האקטיבציה:

$$x_i = f(x_{in_i})$$

שלח את הסיגנל לשכבת Y.

2.3.3. בדוק התכנסות:

אם וקטור שכבת X ווקטור שכבת Y הגיעו לשיווי משקל- עצור. אחרת, המשך.

### נוכח כי האלגוריתם מסתיים

הקדמה לחישובים:

על מנת להוכיח שהאלגוריתם מסתיים נדמה את האלגוריתם לפונקציה ונראה שהפונקציה מתכנסת תמיד ולכן גם האלגוריתם מתכנס ומסתיים.

פונקציית האקטיבציה הינה הפונקציה הבאה:

$$L = -0.5(xWy^t + yW^t x^t)$$

מאחר ש  $xWy^t$  ו  $yW^t x^t$  הם סקלרים, ושיחלוף של סקלר זה הסקלר עצמו נקבל:

$$L = -xWy^t = -\sum_{j=1}^m \sum_{i=1}^n x_i w_{ij} y_j$$

נמשיך עם אותה הוכחה של Hopfield.

החסם התחתון של הפונקציה הינו:

$$-\sum_{j=1}^m \sum_{i=1}^n |w_{ij}|$$

פונקציה זו חסומה על ידי קבוע מלמטה, נוכיח שהיא תמיד יורדת וכן לאחר כמות סופית של צעדים היא תתכנס לגבול שלה.

הקיבולת זיכרון של רשת bam הינה:

$$\min(n, m)$$

כאשר n הוא המימד של אחד מהוקטורים בזוג ו m הוא של השני.

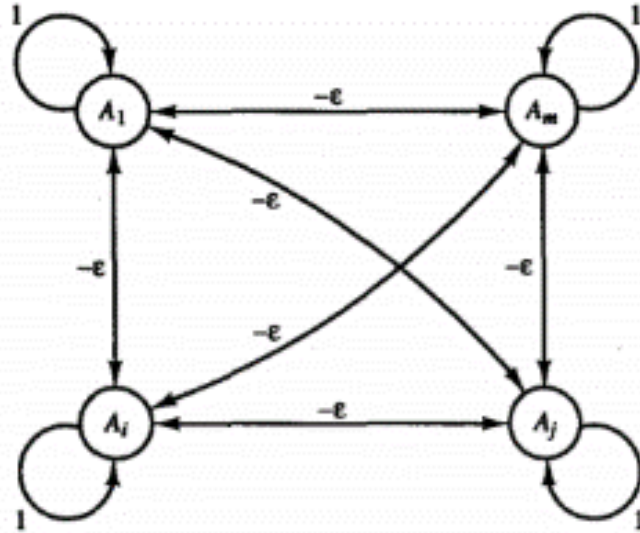
### שימושים

הרשת זוכרת זוגות של וקטורים (אסוציאציות), לדוגמא:

(תמונה של סבתא, ריח של עוגת הפאי שלה), בהינתן וקטור, לדוגמא, תמונה לא ברורה של סבתא, והכנסתו לרשת, נקבל את ריח עוגת הפאי שלה.

עמודים בספר: פרק 3.5

רשת maxnet הינה רשת של נירונים שבה כל נירון מחובר לכולם וגם לעצמו, משקולת של נירון לעצמו היא 1, ובין נירונים שונים היא  $-\epsilon$ , כאשר  $0 < \epsilon < \frac{1}{m}$  ו  $m$  הוא מספר הנירונים



**אלגוריתם**

1. אתחול המשקולות ואקטיבציה, כאשר אקטיבציה של נירון  $i$  שווה לביט  $i$  של וקטור קלט.
2. כל עוד התנאי לא מתקיים, חזור על השלבים הבאים:
  - 2.1. עדכן את האקטיבציה של כל נירון להיות:

$$a_j(new) = f \left[ a_j(old) - \epsilon \sum_{k \neq j} a_k(old) \right]$$

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{כאשר}$$

שמור את האקטיבציות לאיטרציה הבאה .2.2

$$a_j(old) = a_j(new)$$

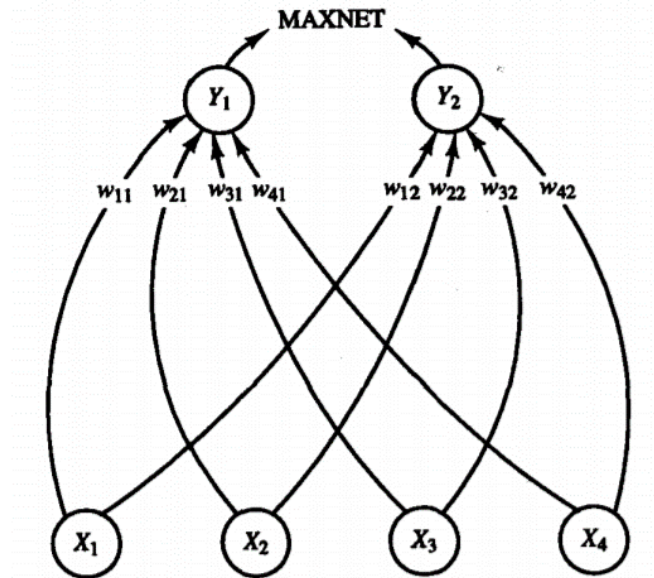
2.3 אם יש מקסימום נירון אחד שערך האקטיבציה שלו הוא שונה מ 0, עצור. אחרת, המשך.

עמודים בספר : פרק 4.1.1

זוהי רשת עם שתי שכבות נוירונים, הנוירונים מחוברים משכבה X ל Y ולא בכיוון ההפוך. כל נוירון בשכבה Y מחובר לרשת Hamming מסווגת את המידע בהתאם לסבירות המקסימלית, כלומר היא מקבלת וקטור ובודקת כמה הוא דומה לכל אחד מווקטורי הזיכרון ומשייך אותו לווקטור הזיכרון שהכי דומה לו.

### Hamming distance

זוהי שיטה למדידת מרחק בין וקטורים, החישוב מתבצע על ידי ספירת הביטים השונים בין שני וקטורים, ככל שהמספר גדול יותר כך הווקטורים רחוקים יותר.



### אלגוריתם

1. על מנת לאחסן את ווקטורי הזיכרון, נאתחל את המשקולות להיות:

$$w_{ij} = \frac{e_i(j)}{2}$$

כאשר  $e_i(j)$  מייצג את הביט ה  $i$  מווקטור  $j$

$$bias = \frac{n}{2}$$

2. עבור כל וקטור קלט בצע את השלבים הבאים:

2.1. חשב את הקלט של נוירון  $Y_j$ :

$$y_{in_j} = b_j + \sum_i x_i w_{ij}$$

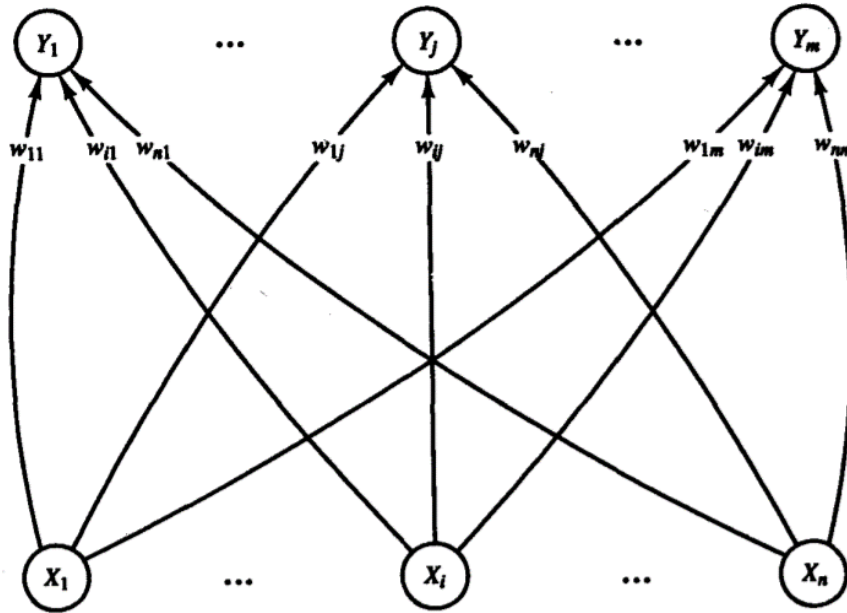
2.2. אתחל את האקטיבציות של רשת maxnet להיות

$$y_j(0) = y_{in_j}$$

2.3. רשת maxnet מוצאת את וקטור הזיכרון הדומה ביותר לווקטור הקלט.

עמודים בספר: 4.1.3

רשת זו עוזרת לעשות סיווגים וניתוח על התפלגות המידע בכך שהיא לוקחת ניוירונים המחוברים בסדר טופולוגי מסוים וממקמת אותם על המידע, לכל ניוירון ישנם שני מאפיינים, אחד שכניו בסדר הטופולוגי והשני מערך של משקלים המייצג את קבוצת המידע שהוא הנציג שלה. בהינתן מידע אנחנו רוצים לדעת התפלגות ומקבצים של חלקי מידע שקרובים אחד לשני, בנוסף נרצה לשמור על הסדר הטופולוגי בתוך המקבצים ובין המקבצים כפי שהיה קיים במידע המקורי.



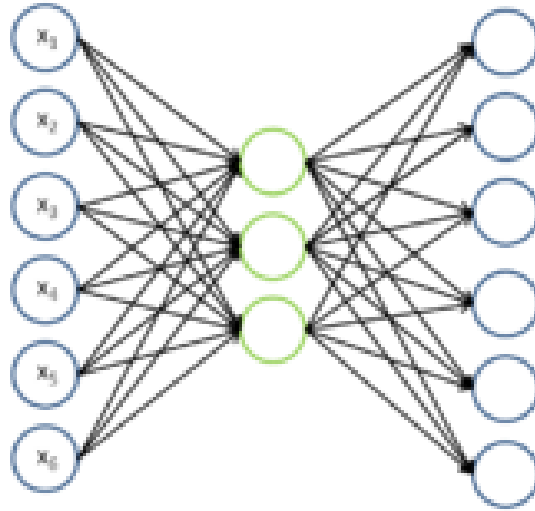
### אלגוריתם

1. בחר נקודה של מידע מהמאגר
2. מצא את הניירון הקרוב ביותר לנקודת המידע שנבחרה (על ידי נוסחת מרחק בין שתי נקודות)
3. תקרב את המקבץ לדגימה שנבחרה על ידי הנוסחה הבאה:
 
$$w_{new} = w_{old} + \alpha(x - w_{old})$$
4. כאשר  $0 < \alpha < 1$  הינה הקצב לימוד
5. עדכן את המרחקים של השכנים (עד דרגה מסוימת) לפי אותה נוסחה שבשלב 3
6. אחרי כמות מסוימת של איטרציות הקטן את  $\alpha$  וכן את רדיוס השכנים לעדכון.
7. כאשר עוצרים אזי בחירה רנדומלית של דגימה נופלת בהסתברות שווה על כל אחד מהמקבצים (=equiprobable)
- 8.

עמודים בספר: פרק 4.2

## Counter Propagation Network

זוהי רשת רב שכבתית המבוססת על קומבינציה של קלט, clustering ופלט.  
רשתות אלו יכולות לשמש לדחיסת מידע, שערך פונקציות ולשיוך דפוסים (אסוציאציות).  
השכבה הנסתרת של הרשת היא רשת Kohonen.



עמודים בספר : פרק 4.4



## Sparse Distributed Memory

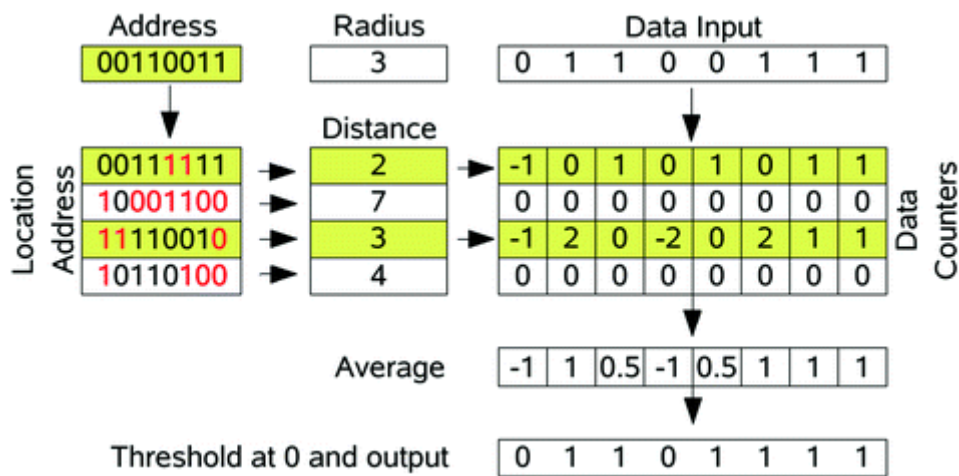
בהינתן מרחב מידע שאי אפשר להחזיק את כולו בגלל גודלו, אזי נרצה להחזיק  $m$  נקודות במרחב שיקראו hard location שבעזרתם נוכל לתאר ולקבל את כל מרחב המידע.

בשיעור ראינו שניתן להשתמש בזה על מנת להחזיק כמות כתובות גדולה יותר ממה שהמחשב מסוגל להכיל, וזאת על ידי האלגוריתם הבא:

בהינתן כתובת וירטואלית נמצא את כל הכתובות האמיתיות שהן במרחק hamming מסוים מכתובת זו. בכל כתובת נמצא וקטור של מספרים שלמים, עבור כל הכתובות שהן במרחק המתאים נחבר את ווקטורי המידע שהן מייצגות לכדי וקטור אחד. בווקטור התוצאה נשנה כל ביט באופן הבא:

אם התוכן של הביט גדול מאפס נחליף אותו ב 1, אחרת נחליף אותו ב 0.

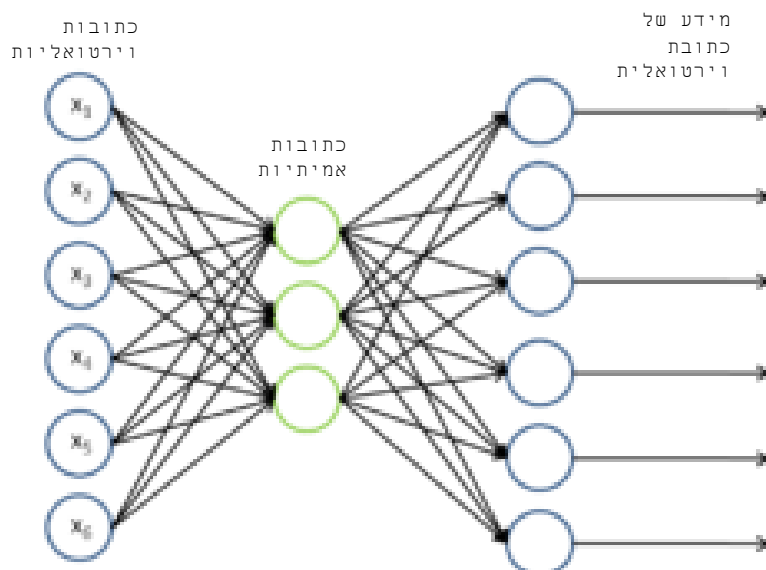
וקטור האפסים והאחדות שהתקבל הינו ווקטור המידע ששייך לכתובת הווירטואלית.



מקור: [https://link.springer.com/chapter/10.1007/978-94-017-8832-8\\_40](https://link.springer.com/chapter/10.1007/978-94-017-8832-8_40)

הכתובות הווירטואליות נכנסות כקלט לשכבה הראשונה, לכל נוירון בשכבה השנייה המשקולות הנכנסות אליה מייצגות את כתובתו בזיכרון האמיתי.

המשקולות שיוצאות מנוירון בשכבה האמצעית מייצגות את המידע שהוא מצביע עליו.



## **TSDM**

בתוך ה hard location ישנן כתובות /ניירונים עם מידע אמין וישנן כתובות/ניירונים עם מידע פחות אמין. אחרי כל פעימת זמן נעשה רעש למידע של כתובת /ניירון ואחרי כמות פעמים מסויימת כתובת /ניירון זה "יכבו"

עמודים בספר : אין

כאשר שני נוירונים המקושרים אחד לשני באופן דו כיווני פועלים באותו הרגע, אז נרצה שהקשר ביניהם יתחזק

$$w_{ij} = \frac{1}{p} \sum_{k=1}^p x_i^k x_j^k$$

כאשר  $p$  הוא מספר הנוירונים ברשת,  $x_i^k$  – הכוונה הקלט ה  $k$  בנוירון  $i$ .

קישור : [https://en.wikipedia.org/wiki/Hebbian\\_theory](https://en.wikipedia.org/wiki/Hebbian_theory)