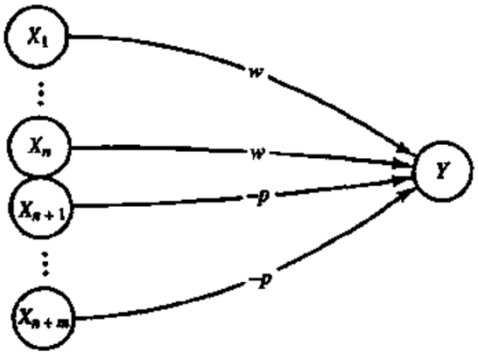
**Mc-Cullough Pitts Neuron:**




הקדמה:

- פונקציית האקטיבציה היא בינארית, כלומר כל ניורון יורה 0 או 1.
- הניורונים קשירים, מכוונים וממושקלים.
- משקולת היא חיובית או שלילית, כל המשקולות החיוביות הנכנסות לניורון מסויים הן בעלי אותו משקל, כנ"ל לגבי השליליות.
- לכל ניורון יש threshold קבוע כך שאם הקלט של הניורון גדול שווה מ$t$ הניורון ירה 1 ואחרת 0.
- קובעים את הthreshold כך שאם משקולת שלילית נכנסת אליו סכום ה$x_i w_i$ בהכרח יהיה קטן מt.

ארכיטקטורה: כל ניורון ברשת מקבל סיגנל ממספר ניורונים אחרים. כל חיבור הוא ממושקל, יש $n$ משקלים חיוביים (זהים) ו$m$ משקלים שליליים (זהים).
הקלט של כל ניורון (למעט שכבת הקלט) הוא $\sum x_i w_i$.
פונקציית אקטיבציה: $f(y_{in}) = 1 \; if \; y_{in} \geq \theta, \; 0 \; elsewere$.

על ידי רשת נוירונים נוכל לבנות כל פונקציה לוגית. שהרי ניתן ליצור על ידי נוירון בודד את הפונקציות AND, OR, NOT וכך ניתן לבנות רשת בעלת 3 שכבות וכל שכבה תיהיה אחראית על פונקציה לוגית אחרת.

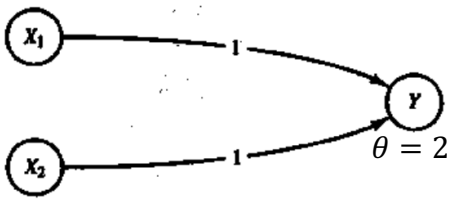



**Figure 1.14** A McCulloch-Pitts neuron to perform the logical AND function.

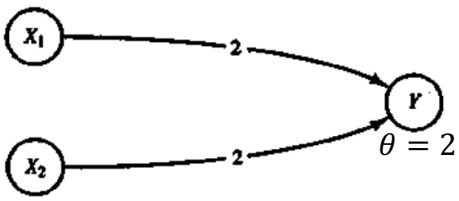
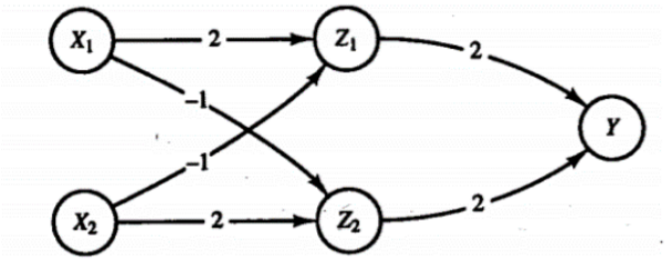



**Figure 1.15** A McCulloch-Pitts neuron to perform the logical OR function.

לכן ניתן לממש XOR שמשוואתו בצורת DNF היא: $X_1 \oplus X_2 = (X_1 \; and \; not \; X_2) \; or \; (X_2 \; and \; not \; X_1)$




משפט:
כל פונקציה לוגית (בוליאנית) עם כל מספר של משתנים ניתן לייצג על ידי רשת McCulloch-Pitts.
הוכחה:
הצגת DNF: $(x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4)$.
כל כניסה שיש לה בהצגת DNF שלילה תכנס בתחילה לניורון שאחראי על פונקציית הNOT.
כל זוג תוצאות (אילו שנכנסו לניורון של NOT ואלו שלא) יכנסו יחד לניורון של AND וכל התוצאות האלו יכנסו בזוגות לניורון OR ובהתאם לthreshold נקבל 1 או 0.
מכאן קיבלנו שניתן ליצור כל פונקציה בולינאית על ידי 3 שכבות, כלומר שכבת NOT, שכבת AND ושכבת OR.

שימושים: פונקציות לוגיות, סיווג (classification)

נקודות למבחן:

- אלגוריתם לא לומד.
- אלגוריתם classification.
- ניורון בודד יכול לפתור רק בעיות שניתנות להפרדה לינארית.

נכתב ע"י איתי לשובר ©

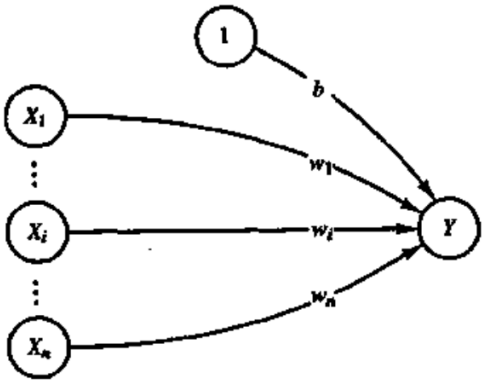
**Perceptron:**

<u>הקדמה:</u>

כלל הלמידה של הPerceptron תחת הנחות מתאימות, מתכנס למשקלים הנכונים, כלומר, המשקולות המאפשרות לרשת לייצר את ערך הפלט הנכון עבור כל אחת מדפוסי הקלט בקבוצת האימון.

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

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




<u>הבדלים בין פרספטרון לנוירון MP :</u>

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

<u>ארכיטקטורה</u>: באיור משמאל.

<u>פונקציית אקטיבציה:</u>

$$f(y_in) = \begin{cases} 1 & \text{if } y_in > \theta \\ 0 & \text{if } -\theta \le y_in \le \theta \\ -1 & \text{if } y_in < -\theta \end{cases}$$

<u>אלגוריתם:</u>

האלגוריתם שניתן כאן מתאים לוקטורי קלט בינאריים או ביפולרים (n- tuples), עם מטרה ביפולרית, $\theta$ קבועה וbias משתנה.

האלגוריתם אינו רגיש במיוחד לערכים ההתחלתיים של המשקולות או לערך קצב הלמידה ($\alpha$).

<u>הערה חשובה:</u> נוסיף ניורון קלט נוסף (שהוא אחראי על הbias) שערכו תמיד 1 והמשקל שיוצא ממנו לניורון הקלט הוא הbias. בצורה זאת נוכל להתייחס ל$\theta$ כ0 ולשנות רק את הbias.

קלט: קבוצה של זוגות סדורים $(s,t)$ – וקטור וlabel (לדוגמה: $(s=[1,0], t=1)$).

0. אתחול רנדומלי של המשקולות והbias (בשביל הפשטות ניתן לאתחל הכל באפסים).
   אתחל את ערך קצב הלמידה להיות $0 < \alpha \le 1$ (בשביל הפשטות ניתן לאתחל $\alpha = 1$).
1. כל עוד לא הגענו לתנאי העצירה (סיבוב שלם ללא טעויות).
   *a.* לכל זוג $s,t$ ששייך לקבוצת האימון:
      *i.* בצע אקטיבציה לניורוני הקלט - $x_i = s_i$.
      *ii.* חשב את הקלט של ניורון הפלט בצורה הבאה: $y_{in} = \sum_i x_i w_i$ (גם ה*bias* נסכם עבור $i = 0$).
      *iii.* עדכון המשקולות (וה*bias* שהוא בעצמו משקולת $w_0$) אם יש שגיאה:
      אם $f(y_{in}) \neq t$:
      $$w_i(new) = w_i(old) + \alpha t x_i$$
      $$b_i(new) = b_i(old) + \alpha t$$
2. אם שום משקולת לא עודכנה במהלך סיבוב שלם (כלומר מעבר על כל הזוגות) נעצור, אחרת חזור לשלב 1.

**הערה:** נשים לב שבמהלך עדכון המשקולות, רק משקולת שה$x_i = 1$ תתעדכן.
בנוסף, עדכון מתבצע רק כאשר האלגוריתם מוציא תשובה שגויה.

<u>משפט ההתכנסות של הפרספטרון:</u>

בהינתן קבוצה P של זוגות קלטים לאימון ברשת הניורונים, אם קיים וקטור משקולות $w^*$ כך ש$f(x(p) \cdot w^*) = t(p)$ מתקיים לכל $p \in P$ אזי עבור כל וקטור משקולות התחלתיים, כלל הלמידה של הפרספטרון יתכנס לוקטור משקולות אשר יביא את התגובות הנכונות עבור כל אחד מהדוגמאות הנלמדות, ויעשה זאת בכמות סופית של צעדים.

<u>הוכחה:</u>

בהינתן סט סופי P של וקטורי אימון, נגדיר: $x(p), t(p),\ \ p = 1,2,..,P$ להיות הוקטור והlabel לכל אחד מהזוגות בקלט, כאשר $t(p)$ הוא 1 או -1.

אם $y \neq t$ (כאשר $y$ זוהי תוצאת פונקציית האקטיבציה) נעדכן את המשקולות בצורה הבאה: $w(new) = w(old) + tx$.

נגדיר $F^+ = \{x \mid t(x) = 1\}, F^- = \{x | t(x) = -1\}$.

נכתב ע"י איתי לשובר ©

נגדיר קבוצת אימון חדשה: $F = F^+ \cup -F^-$ כאשר $-F^- = \{-x | x \in F^-\}$.
כדי לפשט את החישובים האלגבריים, נניח בה"כ ש: $\theta = 0, \alpha = 1$.
קיומו של וקטור משקל $w^*$ עבורו $xw^* < 0 \ (if\ x \in F^-)$ וגם $xw^* > 0 \ (if\ x \in F^+)$ שווה ערך לקיומו של וקטור משקל $w^*$ עבורו $xw^* > 0 \ (if\ x \in F)$.
כך ה*label*ים של קבוצת האימון החדשה הם 1. אם תגובת הרשת שגויה עבור כל אחד מוקטורי האימון שבקלט, המשקלים יתעדכנו ע"י הנוסחה: $w(new) = w(old) + x$.
כאמור, אנו מניחים שקבוצת האימון שונתה כך שכל ה*label*ים הם 1. כדי להשיג זאת, יש להפוך את הסימן של כל הרכיבים (כולל רכיב ה*bias*) עבור כל וקטורי הקלט שעבורם המטרה הייתה במקור *-1*.
כעת, נתבונן על רצף וקטורי האימון שבקלט שעבורם היה שינוי במשקל (כלומר, כל אותם וקטורים שהרשת הוציאה לגביהם תשובה שגויה). נרצה להראות שרצף זה הוא סופי.
נגדיר את וקטור המשקולות ההתחלתי להיות $w(0)$, האותו אופן נגדיר את וקטור המשקולות החדש (לאחר שינוי אחד) להיות $w(1)$, וכו.
יהי $x(0)$ וקטור האימון הראשון שהרשת הוציאה לגביו תשובה שגויה, לכן:
$w(1) = w(0) + x(0)$ (כאשר ההנחה היא ש$x(0) \cdot w(0) \leq 0$).
אם תופיע טעות נוספת, נקבל: $w(2) = w(1) + x(1)$ וכן הלאה.
בכל שלב, נניח שלב $k$, של התהליך, המשקולות משתנות אמ"מ המשקולות הנוכחיים נכשלים לספק תשובה נכונה (1) עבור וקטור האימון הנוכחי שבקלט, במילים אחרות אם $x(k-1) \cdot w(k-1) \leq 0$.
שילוב כל שינויי המשקל העוקבים נותן: $w(k) = w(0) + x(0) + x(1) + x(2) + \cdots + x(k-1)$.
נראה ש$k$ חסום.

1. יהי $w^*$ וקטור משקולות אשר מקיים $x \cdot w^* > 0$ לכל $x \in F$.
יהי $m = \min_{x \in F}\{x \cdot w^*\}$, נקבל ש: $w(k) \cdot w^* = [w(0) + x(0) + x(1) + \cdots + x(k-1)] \cdot w^* \geq w(0) \cdot w^* + km$.
לפי משפט קושי-שוורץ: $(a \cdot b)^2 \leq \left||a|\right|^2 \left||b|\right|^2$ , נציב $a = w(k), b = w^*$, נעביר אגפים ונקבל
$\left||w(k)|\right|^2 \geq \frac{(w(k) \cdot w^*)^2}{\left||w^*|\right|^2} \geq \frac{(w(0) \cdot w^* + km)^2}{\left||w^*|\right|^2}$ (א"ש ראשון – העברת אגפים של קושי שוורץ, א"ש שני – מהמשוואה 2 שורות למעלה).
2. ידוע ש$w(k) = w(k-1) + x(k-1)$ ויחדד עם ההנחה ש$w(k-1) \cdot x(k-1) \leq 0$ נקבל ע"י אלגברה פשוטה ש:

$$\begin{aligned}\left||w(k)|\right|^2 &= \left||w(k-1) + x(k-1)|\right|^2 \\ &= \left||w(k-1)|\right|^2 + 2x(k-1) \cdot w(k-1) + \left||x(k-1)|\right|^2 \\ &\leq \left||w(k-1)|\right|^2 + \left||x(k-1)|\right|^2\end{aligned}$$

יהי $M = \max_{x \in F}\{\left||x|\right|^2\}$ אזי:

$$\begin{aligned}\left||w(k)|\right|^2 &\leq \left||w(k-1)|\right|^2 + \left||x(k-1)|\right|^2 \\ &\leq \left||w(k-2)|\right|^2 + \left||x(k-2)|\right|^2 + \left||x(k-1)|\right|^2 \\ &\leq \left||w(k-3)|\right|^2 + \left||x(k-3)|\right|^2 + \left||x(k-2)|\right|^2 + \left||x(k-1)|\right|^2 \\ &\leq \left||w(0)|\right|^2 + \left||x(k0)|\right|^2 + \left||x(1)|\right|^2 + \cdots + \left||x(k-1)|\right|^2 \\ &\leq \left||w(0)|\right|^2 + kM\end{aligned}$$

משילוב שני החסמים נקבל: $\frac{(w(0) \cdot w^* + km)^2}{\left||w^*|\right|^2} \leq \left||w(k)|\right|^2 \leq \left||w(0)|\right|^2 + kM$ ,
ומכאן ש$\frac{(w(0) \cdot w^* + km)^2}{\left||w^*|\right|^2} \leq \left||w(0)|\right|^2 + kM$.
עתה, נניח בה"כ ש$w(0) = 0$ ומכאן: $\frac{(km)^2}{\left||w^*|\right|^2} \leq kM \rightarrow k \leq \frac{M\left||w^*|\right|^2}{m^2}$.
סה"כ קיבלנו שk חסום מלמעלה ולכן מספר התיקונים לוקטור המשקולות סופי ומכאן שהאלגוריתם תמיד מסתיים.

<u>שימושים:</u> פונקציות לוגיות, סיווג (classification)

<u>נקודות למבחן</u>

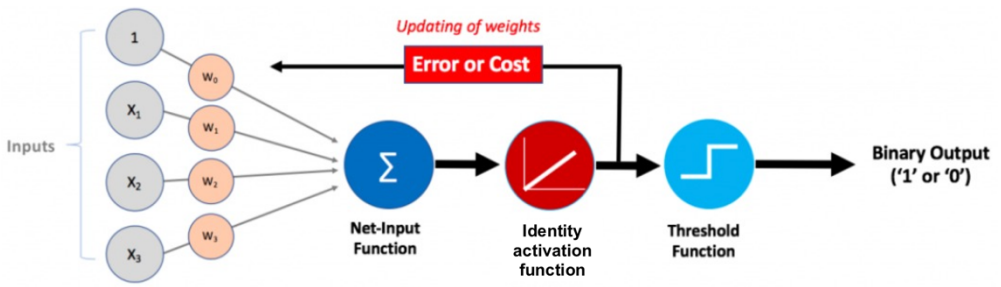
- מה קורה שמנסים לממש בעיית xor באמצעות Perceptron? האלגוריתם לא יעצור.
- ניורון בודד יכול לפתור רק בעיות שניתנות להפרדה לינארית.

נכתב ע"י איתי לשובר ©

**Adaline:**

הקדמה:

- Adaline בדרך כלל משתמש באקטיבציית bipolar (1 או -1) עבור הסיגנלים שהוא קולט.
- המשקולות של הרשת משתנים תוך כדי הלמידה.
- בדומה לPerceptron גם לAdaline יש bias שמתנהג כמו המשקולות כאשר ניורון הקלט שלו תמיד עם ערך 1.
- באופן כללי, הAdaline מתאמן על ידי delta rule, שידוע גם כ least mean squares (LMS).
- הAdaline הוא מקרה מיוחד שבו יש ניורון בודד בoutput.
- במהלך האימון, פונקציית האקטיבציה של ניורון הפלט היא פונקציית הזהות.
- קצב הלמידה ממנן את הLMS בין פונקציית האקטיבציה לבין פונקציית הtreshold שמחזירה 1 או 0, מה מאפשר לרשת להמשיך ללמוד את כל הקלטים בקבוצת הtrain אפילו אחרי שהתשובה הנכונה חוזרת עבור חלק מהדוגמאות בקבוצה.
- בסיום האימון, אם הרשת מיועדת לפתרון בעיית classification שבה הפלט הדרוש הוא 1 או 0 (1 או -1 בbipolar), פונקציית threshold מתבצעת.
- אם הקלט של ניורון הAdaline גדול מ0 הניורון יפלוט 1, 0 אחרת.
- Adaline מסוגל למדל כל קלט שניתן להפריד לינארית.
- האלגוריתם תמיד עוצר.




ארכיטקטורה:

Adaline הוא ניורון בודד שמקבל את הקלט שלו ממספר יחידות שונות.
בנוסף הוא מקבל קלט מיחידה שערכה תמיד 1 כדי שהbias יוכל להתעדכן גם הוא במהלך האימון.

אלגוריתם:

0. אתחול המשקולות לערך רנדומלי קרוב ל0.
   אתחול קצב הלמידה $\alpha$.
1. כל עוד תנאי העצירה לא מתקיים:
   *a.* לכל זוג ביפולארי $s, t$ ששייך לקבוצת האימון:
      *i.* בצע אקטיבציה לניורוני הקלט: $x_i = s_i$.
      *ii.* חשב את הקלט של ניורון הפלט בצורה הבאה: $y_{in} = \sum_i x_i w_i$ (גם הbias נסכם עבור $i = 0$).
      *iii.* עדכון המשקולות (והbias שהוא בעצמו משקולת $w_0$):
$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$b_i(new) = b_i(old) + \alpha(t - y_{in})$$
2. נבדוק האם תנאי העצירה מתקיים:
   אם השינוי הגדול ביותר עבור משקולת מסויימת היה קטן מ$\epsilon$ (מספר קטן כלשהו) עצור.
   אחרת חזור לשלב 1.

**הערה:** יש משמעות גדולה לבחירת הערך של $\alpha$ – אם הערך גדול מדי, הלמידה עלולה שלא להתכנס לעולם.
אם הערך קטן מדי, הלמידה תיהיה איטית בצורה משמעותית.

שימושים: סיווג (classification)

נקודות למבחן

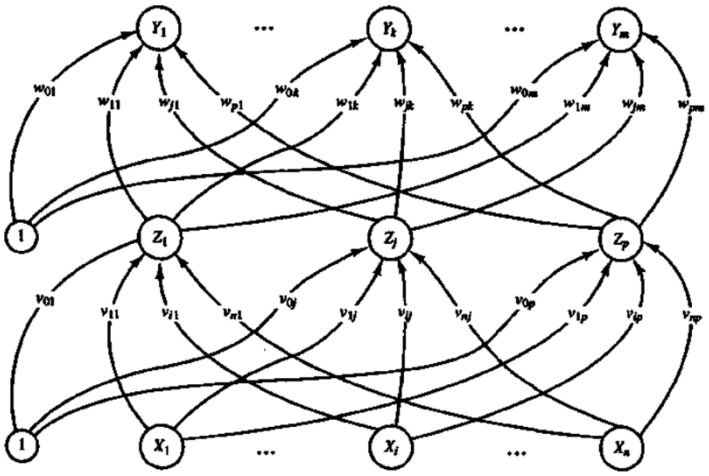
- מה קורה שמנסים לממש בעיית xor באמצעות Adaline? האלגוריתם יעצור ויחזיר תשובה שגוייה.
- ניורון בודד יכול לפתור רק בעיות שניתנות להפרדה לינארית.

נכתב ע"י איתי לשובר ©

**Feed-forward Networks and BackPropogation:**

הקדמה:

- שיטת האימון backpropogation, היא פשוט gradiant descent למזעור השגיאה הכוללת בריבוע של הפלט המחושב על ידי הרשת.
- ניתן למצוא יישומים המשתמשים ברשתות כאלה כמעט בכל תחום המשתמש ברשתות ניורונים לבעיות הכרוכות במיפוי סט נתון של קלטים ליעדים מסויימים (רשתות המשתמשות באימון מפוקח).
- כפי שקורה ברוב הרשתות העצביות, המטרה היא לאמן את הרשת להשגת איזון בין היכולת להגיב נכון לדפוסי הקלט המשמשים לאימון (שינון) לבין היכולת לתת תגובות סבירות (טובות) לקלט דומה, אבל לא זהה, לזה המשמש באימון.
- אימון של רשת על ידי backpropogation כולל שלושה שלבים: הזנה קדימה (feed forward) של דפוסי האימון בקלט, חישוב ו backpropogation של השגיאה הקשורה, ותיקון המשקולות.
- לאחר האימון, הרשת עוסקת רק בחישובים של שלב הfeed forward.
- גם אם האימון איטי, רשת מאומנת יכולה לייצר את התפוקה שלה במהירות רבה.
- למרות שרשת חד-שכבתית מוגבלת מאוד במיפויים שהיא יכולה ללמוד, רשת רב-שכבתית (עם שכבה נסתרת אחת או יותר) יכולה ללמוד כל מיפוי רציף.
- יותר משכבה נסתרת אחת עשויה להועיל עבור יישומים מסוימים, אך מספיקה שכבה נסתרת אחת.




ארכיטקטורה:

כל נוירון בכל אחת מהשכבות מחובר לכל אחד מהנוירונים בשכבה הבאה.

לשכבות הנסתרות ולשכבת הפלט יש bias.

הbias פועל כמשקולת שפונקציית האקטיבציה שלה היא תמיד 1.

באיור משמאל מוצג רק כיוון זרימת המידע עבור שלב הfeed forward של הפעולה. במהלך שלב הbackpropogation של הלמידה, אותות נשלחים בכיוון הפוך.

הקדמה לאלגוריתם:

במהלך הfeed forward כל יחידת קלט $(X_i)$ מקבלת קלט ושולחת אותו לכל אחד מהניורונים $Z_1, \dots, Z_p$ בשכבה החבויה. כל נוירון חבוי מחשב את האקטיבציה שלו ושולח את הסיגנל שלו $(z_j)$ לכל אחד מניורוני הפלט.

כל ניורון פלט $(Y_k)$ מחשב את האקטיבציה שלו $(y_k)$ כדי ליצור את תשובת הרשת עעבור הקלט הנתון.

במהלך האימון, כל ניורון פלט משווה בין האקטיבציה המחושבת שלו $y_k$ לבין ערך המטרה $t_k$ (הlabel המקורי של הקלט) כדי לקבוע את הטעות של הניורון.

בהתבסס על טעות זו, הפקטור $\delta_k$ $(k \in [1,m])$ מחושב. $\delta_k$ משמש להפצת הטעות של ניורון הפלט $Y_k$ בחזרה לכל הניורונים בשכבה הקודמת (הניורונים בשכבה החבויה שמחוברים לאותו $Y_k$), בהמשך $\delta_k$ משמש גם כדי לעדכן את המשקולות בין שכבת הפלט לבין השכבה החבויה.

באופן דומה, הפקטור $\delta_j$ $(j \in [1,p])$ מחושב לכל ניורון חבוי $Z_j$. כאן, אין צורך להפיץ את השגיאה בחזרה לשכבת הקלט, אך $\delta_j$ משמש לעדכון המשקולות בין השכבה החבויה לשכבת הקלט.

לאחר שנקבעו כל הפקטורים $\delta$, המשקלים בכל השכבות מתוקנים במקביל.

תיקון המשקולת $w_{jk}$ (בין הניורון בשכבה החבויה $Z_j$ לבין הניורון בשכבת הפלט $Y_k$) מבוסס על הפקטור $\delta_k$ ועל האקטיבציה $z_j$ של הניורון החבוי $Z_j$.

תיקון המשקולת $v_{ij}$ (בין ניורון הקלט $X_i$ לבין הניורון החבוי $Z_j$) מבוסס על הפקטור $\delta_j$ ועל האקטיבציה $x_i$ של ניורון הקלט.

סימונים:

$\boldsymbol{X}$ - וקטור קלט בשלב האימון: $X = (x_1, \dots x_i, \dots, x_n)$

$\boldsymbol{t}$ - וקטור המטרה: $t = (t_1, \dots, t_k, \dots, t_m)$

$\boldsymbol{\delta_k}$ - חלק של תיקון המשקל עבור $w_{jk}$ הנובע מטעות ביחידת הפלט $Y_k$. כמו כן, זה מציין את המידע על הטעות ביחידה $Y_k$ שמופצת בחזרה לניורונים בשכבה החבויה המזינים את $Y_k$.

$\boldsymbol{\delta_j}$ - חלק של תיקון המשקל עבור $v_{ij}$ הנובע מהbackpropogation של הטעות הנשלח משכבת הפלט לניורון החבוי $Z_j$.

$\boldsymbol{\alpha}$ – קצב הלימוד.

$\boldsymbol{X_i}$ – ניורון הקלט ה$i$. עבור ניורוני הקלט, סיגנל הקלט והפלט של כל ניורון הוא זהה ומסוממן ב$x_i$

נכתב ע"י איתי לשובר ©

(כלומר פונקציית האקטיבציה היא פונקציית הזהות)

$\boldsymbol{v_{0j}}$ – הbias של הניורון החבוי ה$j$.

$\boldsymbol{Z_j}$ –הניורון החבוי ה$j$:

הקלט לניורון $Z_j$ המסומן ב$z_{in_j}$ הוא: $z_{in_j} = v_{0j} + \sum_i x_i v_{ij}$.

הפלט של ניורון $Z_j$ המסומן ב$z_j$ הוא: $z_j = f(z_{in_j})$.

$\boldsymbol{w_{0k}}$ – הbias של ניורון הפלט ה$k$.

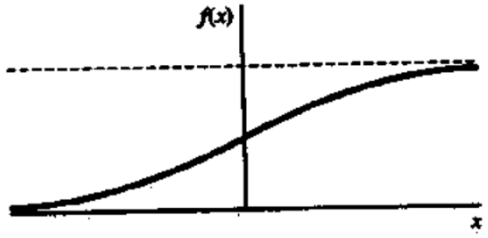
$\boldsymbol{Y_k}$ – ניורון הפלט ה$k$:

הקלט לניורון $Y_k$ המסומן ב$y_{in_j}$ הוא: $y_{in_j} = w_{0k} + \sum_j z_j w_{jk}$.

הפלט של ניורון$Y_k$ המסומן ב$z_j$ הוא: $y_k = f(y_{in_j})$.

<u>פונקציית אקטיבציה:</u>

פונקציית האקטיבציה עבור רשת backpropogation צריך לקיים מספר מאפיינים חשובים:
היא צריכה להיות רציפה, ניתנת להפרדה ומונוטונית לא יורדת.
בנוסף, כדי להשיג יעילות חישובית, רצוי שהנגזרת של הפונקצייה תהיה קלה לחישוב.
אחת מפונקציות האקטיבציה האופייניות ביותר היא פונקציית sigmoid,
מחזירה מספר בטווח $(0,1)$ ומוגדרת: $f(x) = \frac{1}{1+e^{-x}}$.
כאשר הנגזרת שלה היא: $f'(x) = f(x)[1 - f(x)]$.




<u>אלגוריתם:</u>

0. אתחול המשקלים (לערך רנדומלי קטן וקרוב ל0).
1. כל עוד תנאי העצירה לא מתקיים:
   a. לכל זוג בקבוצת האימון (וקטור קלט $x$ ומטרה $t$):

**Feed-Forward:**

i. כל ניורון קלט $X_i$ $\forall i \in [1,n]$ מקבל את סיגנל הקלט $x_i$ ומשגר אותו לכל אחדד מהניורונים בשכבה החבויה.

ii. כל ניורון $Z_j$ $\forall j \in [1,p]$ בשכבה החבויה סוכם את הקלט שקיבל $z_{in_j} = v_{0j} + \sum_i x_i v_{ij}$,
מפעיל את פונקציית האקטיבציה $z_j = f(z_{in_j})$ ושולח את התוצאה לכל הניורונים בשכבת הפלט.

iii. כל ניורון פלט $Y_k$ $\forall k \in [1,m]$ סוכם את הקלט שקיבל $y_{in_j} = w_{0k} + \sum_j z_j w_{jk}$
ומפעיל את פונקציית האקטיבציה $y_k = f(y_{in_j})$.

**Backpropagation of error:**

iv. כל ניורון פלט $Y_k$ מקבל את המטרה $t$ המתאימה לו מוקטור האימון,
מחשב את הטעות שלו $\delta_k = (t_k - y_k)f'(y_{in_k})$,
מחשב את תיקון המשקולת (המשמש לעדכון $w_{jk}$ בהמשך) $\Delta w_{jk} = \alpha \delta_k z_j$,
מחשב את תיקון הbias (המשמש לעדכון $w_{0k}$ בהמשך) $\Delta w_{0k} = \alpha \delta_k$,
ושולח את $\delta_k$ לניורונים בשכבה הקודמת (השכבה החבויה).

v. כל ניורון חבוי $Z_k$ סוכם את קלטי ה$\delta$ שלו (משכבת הפלט): $\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}$,
מכפיל בנגזרת של פונקציית האקטיבציה שלו כדי לחשב את הטעות: $\delta_j = \delta_{in_j} f'(z_{in_j})$,
מחשב את תיקון המשקולות שלו (המשמש לעדכון $v_{ij}$ בהמשך): $\Delta v_{ij} = \alpha \delta_j x_i$
ומחשב את תיקון הbias (המשמש לעדכון $v_{0j}$ בהמשך) $\Delta v_{0j} = \alpha \delta_j$,

**Update weights and biases:**

vi. כל ניורון פלט $Y_k$ מעדכן את הbias ואת המשקולות $\forall j \in [1,p]$: $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$.
כל ניורון חבוי $Z_j$ מעדכן את הbias ואת המשקולות $\forall i \in [1,n]$: $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

   b. בדוק את תנאי העצירה.

<u>הערה</u>: הבסיס המתמטי לאלגוריתם הbackpropoation הוא טכניקת האופטימיזציה המכונה gradient descent.
הגרדיאנט (השיפוע) של הפונקציה (במקרה זה, הפונקציה היא הטעות והמשתנים הם משקלי הרשת) נותן את הכיוון שבו הפונקציה גדלה מהר יותר; השלילי של הגרדיאנט נותן את הכיוון שבו הפונקציה יורדת הכי מהר.

נכתב ע"י איתי לשובר ©

Backpropogation בשילוב עם מומנטום: שינוי המשקל הוא בכיוון שהוא שילוב של השיפוע הנוכחי והשיפוע הקודם.
זהו שינוי של gradient descent שיתרונותיו נובעים בעיקר כאשר חלק מנתוני האימון שונים מאוד מרוב הנתונים (ואולי אפילו לא נכונים). רצוי להשתמש בשיעור למידה קטן כדי למנוע שיבוש גדול בכיוון הלמידה כאשר מוצגים צמד דפוסי אימון חריגים מאוד. עם זאת, עדיף גם לשמור על אימונים בקצב מהיר למדי כל עוד נתוני האימונים דומים יחסית.
ההתכנסות לפעמים מהירה יותר אם מוסיפים מומנטום לנוסחאות עדכון המשקל.
על מנת להשתמש במומנטום, יש לשמור משקלים (או עדכוני משקלים) מדפוסי אימון קודמים (אחד או יותר).
לדוגמה, ביישום הפשוט ביותר של backpropogation עם מומנטום, המשקולות החדשים לאימון בשלב $t+1$ מבוססות על המשקולות בשלבי האימון $t$ ו$t-1$.
מומנטום מאפשר לרשת לבצע התאמות משקל גדולות למדי כל עוד התיקונים נמצאים באותו כיוון כללי עבור מספר דפוסים, תוך שימוש בקצב למידה קטן כדי למנוע תגובה גדולה לשגיאה מכל אחד מדפוס האימון.
זה גם מקטין את הסבירות שהרשת תמצא משקלים שהם מינימום מקומי, אך לא גלובלי.

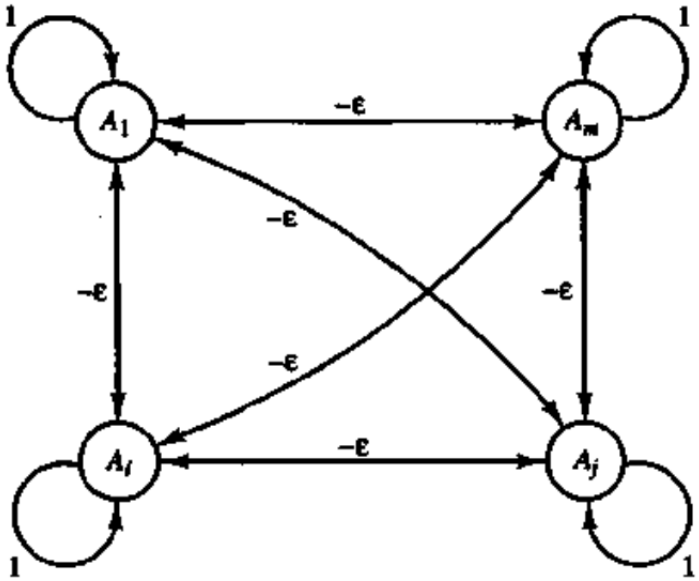
מגבלות האלגוריתם: עלול להיתקע במינימום מקומי כיוון שמשתמש בgradient descent.
זמן הלמידה ארוך (אבל אחרי שהוא לומד הוא מחזיר תוצאות מהר).

שימוש: ייצוג פונקציות בינאריות שלא בהכרח ניתנות להפרדה ליניארית, דחיסת מידע, זיהוי תבניות, אותיות.

**Max Net:**

הקדמה:

- רשת ניורונים המבוססת על תחרות.
- ניתן להשתמש ברשת זו בתור תת-רשת שתפקידה לבחור את הקודקוד שהקלט שלו הוא הגדול ביותר.
- הרשת מורכבת מ$m$ קודקודים כאשר כל קודוקוד מחובר לכל קודקוד אחר ולעצמו.
- כל החיבורים שבין קודקודים שונים ממושקלים באותו משקל $-\epsilon$.
- החיבור שבין קודקוד לעצמו ממושקל ב1.
- אין אלגוריתם אימון עבור רשת ניורונים זו והמשקולות בה קבועים ולא משתנים.




ארכיטקטורה: בתמונה משמאל.

פונקציית האקטיבציה: $f(x) = x \; if \; x \geq 0, \; 0 \; otherwise$

אלגוריתם:

0. אתחול האקטיבציות והמשקלים:
   קבע $0 < \epsilon < \frac{1}{m}$ ($m$ זה מספר הקודקודים ברשת).
   קבע $w_{i,j} = -\epsilon \; of \; i \neq j \; , \quad 1 \; if \; i = j$
1. כל עוד תנאי העצירה לא מתקיים (לא נשאר ניורון עם ערך גדול מ0 בודד):
   a. עדכן את האקטיבציה של כל קודקוד (לכל $j$):
   $$a_j(new) = f[a_j(old) - \epsilon \sum_{k \neq j} a_k(old)]$$
   b. שמור את האקטיבציה לטובת שימוש באיטרציה הבאה: $\forall j \in [1,m] \quad a_j(old) = a_j(new)$.
   c. בדיקת תנאי העצירה: אם ליותר מקודקוד אחד יש ערך שונה מ0 המשך, אחרת עצור.

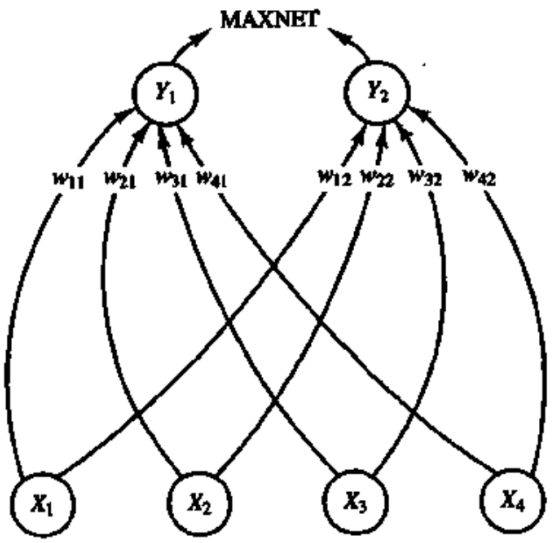
**הערה:** בשלב $a$ הקלט של פונקציית האקטיבציה $f$ הוא בסך הכל כל הקלט שנכנס לקודקוד $A_j$ כולל עצמו.
**הערה:** יש לשלב כמה אמצעי זהירות כדי להתמודד עם המצב שבו לשתי יחידות או יותר יש קלט זהה, מקסימלי.

נכתב ע"י איתי לשובר ©

**Hamming Net:**

<u>הקדמה:</u>

- רשת ניורונים מסוג maximum likelihood classifier, כלומר, רשת המסווגת קלט (וקטור) לוקטור שהכי דומה לו מבין הוקטורים המאוכסנים ברשת.
- הוקטורים המאוכסנים קובעים את המשקולות של הרשת (כך למעשה הרשת מאחסנת אותם).
- המדד לכמה זוג וקטורים דומים הוא $n - Hamming\ Distance$.
- Hammig Distance בין זוג וקטורים הוא כמות הסיביות שיש ביניהן הבדל ב2 הוקטורים ($HD([1,1,1],[1,0,1]) = 1$).
- על ידי הגדרת המשקולות להיות חצי מערך הוקטור המאוחסן והגדרת ערך ה*bias* ל$\frac{n}{2}$, הרשת תמצא את היחידה עם הדוגמה הקרובה ביותר, על ידי מציאת היחידה עם הקלט הגדול ביותר (באמצעות *MaxNet*).
- רשת *Hamming* משתמשת ב*MaxNet* כתת-רשת כדי למצוא את היחידה עם הערך הגבוה ביותר.
- הרשת מורכבת מ*n* קודקודי קלט, כל אחד מהם מחובר ל*m* קודקודי פלט (כאשר *m* זה מספר הדוגמאות המאוחסנות ברשת).
- קודקודי הפלט "מאכילים" (משמשים כקלט) את ה*MaxNet* שמחשב ומוצא את הוקטור המתאים (הדומה ביותר).
- וקטור הקלט והוקטורים המאוחסנים הם ביפולארים.




<u>ארכיטקטורה:</u>

בארכיטקטורה המוצגת משמאל, הקלט הוא וקטור בגודל 4 שיש לסווג לאחד מבין 2 וקטורים מאוחסנים.

<u>אלגוריתם:</u>

סימונים: $n$ – מספר קודקודי הקלט (גודל וקטור הקלט)

$m$ – מספר קודקודי הפלט (כמות הוקטורים המאוחסנים)

$e(j)$ – הוקטור המאוחסן ה$j$.

0. אחסון $m$ הוקטורים ע"י אתחול המשקולות:

$$w_{i,j} = \frac{e_i(j)}{2}, \quad \forall i \in [1,n]\ \forall j \in [1,m]$$

אתחול ה*bias*: $b_j = \frac{n}{2}\ \ \forall j \in [1,m]$

1. לכל וקטור *x*:
   - *a*. חשב את הקלט לניורון $Y_j$: $y_{in_j} = b_j + \sum_{j=1}^{m} x_i w_{ij}$
   - *b*. אתחול האקטיבציה עבור MaxNet: $y_j(0) = y_{in_j}\ \ \forall j \in [1,m]$
   - c. ביצוע אלגוריתם הMaxNet למציאת הוקטור המתאים ביותר לוקטור x.

<u>הערה</u>: למעשה, הפלט של הנוירונים שנכנס לקלט של קודקודי הMaxNet הוא מספר הביטים הזהים בין וקטור x לבין הוקטור המאוחסן $e_j$.

<u>שימוש</u>: בהינתן $m$ וקטורים ביפולארים $e(1),..e(m)$ Hamming Net ימצא את הוקטור הדומה ביותר לוקטור קלט כלשהו x.

<u>נקודות למבחן</u>

- אלגוריתם clustering.

נכתב ע"י איתי לשובר ©

**Kohonen Net:**

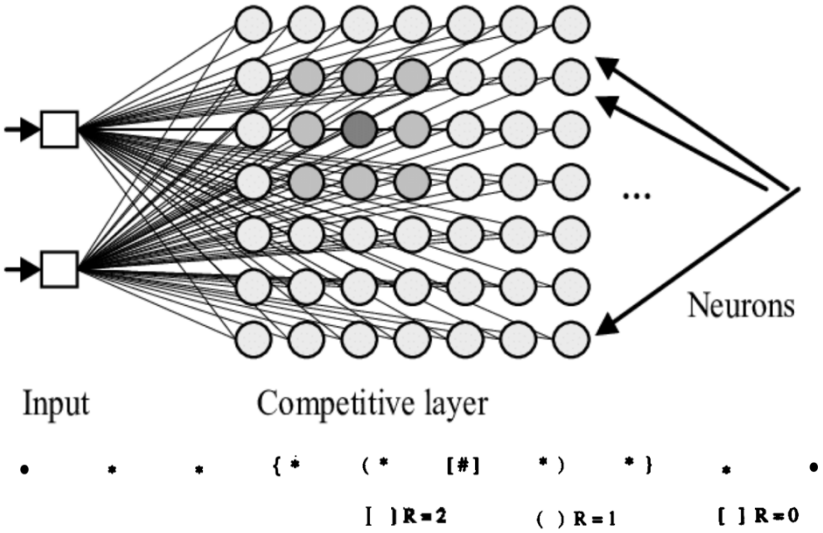
הקדמה – Kohenen self-organizing map:

- רשתות עצביות המתארגנות בעצמן (הנקראות גם מפות משמרות טופולוגיה), מניחות מבנה טופולוגי בין הנוירונים.
- ישנם $m$ ניורונים, המסודרים במערך חד או דו מימדי, אותות הכניסה הם בגודל $n$.
- וקטור המשקל עבור ניורון מסויים משמש דוגמה לדפוסי הקלט הקשורים לאותו ניורון.
- במהלך תהליך הארגון העצמי, הניורון שוקטור המשקל שלו תואם את דפוס הקלט בצורה ההדוקה ביותר (בדרך כלל, המרחק האוקלידי המינימלי בריבוע) נבחר כמנצח.
- הניורון הזוכה והניורונים השכנים שלו (מבחינה טופולוגית) מעדכנות את משקלן.

ארכיטקטורה:

התמונה משמאל היא ארכיטקטורה של Kohonen Net כאשר הניורונים מסודרים במערך דו מדדי.

בתמונה התחתונה מתוארת ארכיטקטורה שבה $m$ הניורונים מסודרים במערל חד ממדי והניורון # הוא הניורון הנבחר.
בדוגמה זו כל הניורונים הנמצאים בטווח רדיוס=2 יעדכנו גם הם את משקולם (יזוזו יחד עם הניורון הנבחר).
בתמונה ניתן לראות שהניורונים שבמרחק רדיוס 1 המסומנים בתוך סוגריים רגילים והניורונים שבמרחק רדיוס 2 המסומנים בתוך סוגריים מסולסלים נבחרו בתור השכנים שמשקלם יתעדכן.




אלגוריתם:

0. אתחול כל המשקולות $w_{ij}$ (ע״י ערכים רנדומלים או ע״י ידע מוקדם על התפלגות הclusterים).
   קביעת הפרמטרים לסדר הטופולוגי (הרדיוס).
   קביעת ערך לקצב הלמידה $\alpha$.
1. כל עוד תנאי העצירה לא מתקיים:
   a. לכל וקטור קלט $x$:
      i. לכל $j \in [1,m]$ חשב: $D(j) = \sum_i (w_{ij} - x_i)^2$.
      ii. מצא את האינדקס $J$ עבורו $D(J)$ מינימלי.
      iii. לכל הניורונים $j$ שהם שכנים (בתוך הרדיוס הנבחר) של הניורון הנבחר $J$ ולכל $i$:
      $$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)]$$
   b. עדכון קצב הלמידה $\alpha$.
   c. עדכון הרדיוס לאחר מספר איטרציות.
   d. בדוק את תנאי העצירה (בדר״כ תנאי העצירה יהיה מספר איטרציות או התכנסות לצורה מסויימת).

במילים פשוטות יותר:

בחירת נקודת מידע מהמאגר, מציאת הנוירון הקרוב ביותר לנקודת המידע שנבחרה (על ידי נוסחת מרחק בין שתי נקודות), קירוב המקבץ לדגימה שנבחרה על ידי עדכון המשקולות, עדכון המרחקים של השכנים לפי אותה נוסחה, הקטנת $\alpha$ ורדיוס השכנים לאחר מספר איטרציות.

שימושים: רשת ניורונים ליצירת מוזיקה על ידי מחשב, פתרון TSP (בעיית הסוכן הנוסע), Character Recognition (זיהוי תווים), Spanning Tree.

נקודות למבחן

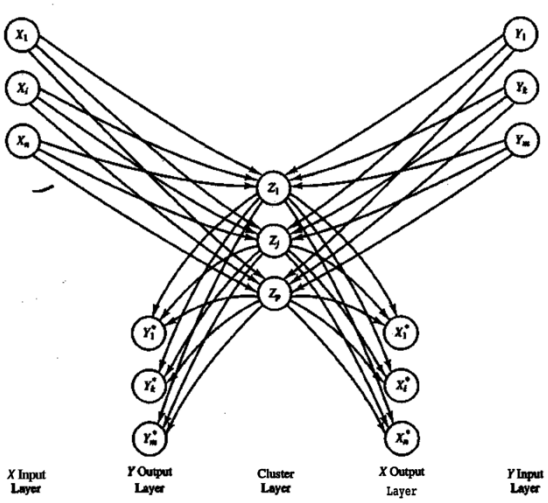
- אלגוריתם clustering.

נכתב ע״י איתי לשובר ©

**Counter Propagation Network:**

<u>הקדמה:</u>

- רשתות רב-שכבתיות המבוססות על שילוב של שכבות קלט, clustering ופלט.
- ניתן להשתמש ברשתות Counter Propogation כדי לדחוס נתונים, לקרב פונקציות או לשייך דפוסים.
- רשתות Counter Propogation מאומנות בשני שלבים.
  - במהלך השלב הראשון, וקטורי הקלט מקובצים לclusterים.
    בהגדרה המקורית, לא הונחה טופולוגיה עבור יחידות הcluster, עם זאת, הוספת טופולוגיה ליניארית יכולה לשפר את ביצועי הרשת.
    הclusterים שנוצרים עשויים להתבסס ע״פ מכפלה סקלרית או ע״פ נורמה אוקלידית.
  - בשלב השני של האימון מותאמים המשקולות מיחידות הcluster ליחידות הפלט כדי לייצר את התגובה הרצויה.




<u>Full Counter Propagation:</u>

מספקת שיטה יעילה לייצוג מספר רב של זוגות וקטורים $x:y$, על ידי בנייה אדפטיבית של טבלת חיפוש (lookup table).

רשת זו מייצרת וקטורי קירוב $x^* : y^*$ המבוססים על קלט של וקטור $x$ (ללא מידע על וקטור $y$ המתאים), או קלט של וקטור $y$ בלבד, או על קלט של זוג וקטורים $x:y$ עם כמה אלמנטים מעוותים או חסרים באחד הוקטורים או בשניהם.

Full Counter Propagation משתמשת בזוגות הוקטורים $x:y$ מקבוצת האימון, כדי ליצור את הclusterים במהלך השלב הראשון של האימון.

בהגדרה המקורית, התחרות בשכבת הcluster (או שכבת Kohonen) בחרה את היחידה שהיה לה את הקלט הגדול ביותר כמנצחת.

<u>שימושים:</u> דחיסת נתונים, שיערוך פונקציות וסיווג ווקטורים (cluster)

<u>נקודות למבחן</u>

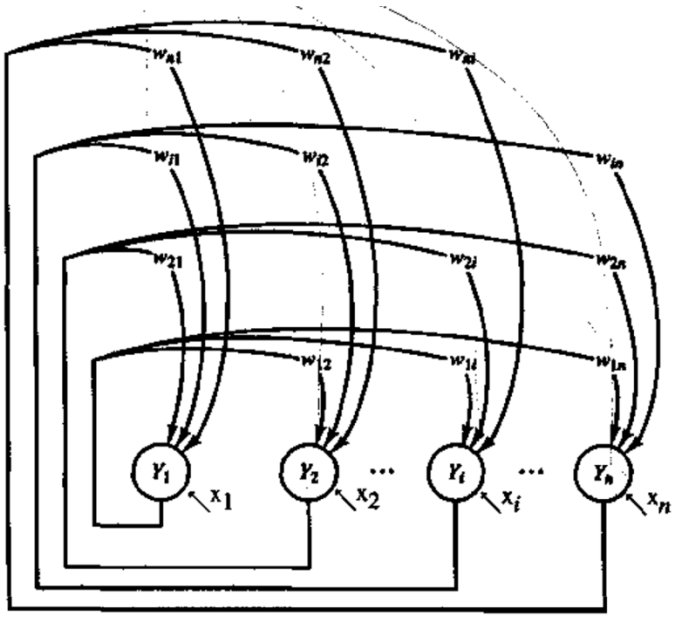
- אלגוריתם clustering.

נכתב ע״י איתי לשובר ©

**Hopfield Network:**

הקדמה:

- רשת אוטו-אסוציאטיבית איטרטיבית.
- הרשת היא רשת ניורונים מחוברת לחלוטין, במובן זה שכל ניורון מחובר לכל ניורון אחר.
- לרשת יש משקלים סימטריים ללא חיבורים עצמיים, כלומר, $w_{ij} = w_{ji}$ וגם $w_{ii} = 0$.
- ברשת זו, רק ניורון אחד מעדכן את הפעלתו בכל פעם (בהתבסס על האות שקיבל מכל אחד מהניורונים האחרים) ובנוסף, כל יחידה ממשיכה לקבל אות חיצוני בנוסף לאות משאר הניורונים ברשת.
- העדכון האסינכרוני של הניורונים מאפשר למצוא פונקציה, המכונת פונקציית אנרגיה, עבור הרשת. קיומה של פונקציה כזו מאפשר לנו להוכיח שהרשת תתכנס לקבוצה יציבה של הפעלות, במקום להתנודד.
- הניסוח המקורי של רשת Hopfield הדיסקרטית הראה את התועלת של הרשת כזיכרון תוכן משוייך (content-addressable memory).




ארכיטקטורה:

רשת ניורונים מחוברת לחלוטין עם משקלים סימטריים וללא חיבורים עצמיים.
בהיותו מחובר במלואו, הפלט של כל נוירון הוא קלט לכל שאר הנוירונים אך לא לניורון עצמו.

אלגוריתם:

0. אתחול המשקלים לאכסון הדפוסים:
   $W = \sum_x x^T x$ ולאחר מכן לעדכן לכל $i$: $w_{ii} = 0$.
1. לכל וקטור קלט $x$:
   a. אתחול האקטיבציה של ניורוני הקלט שיהיו שווים לוקטור $x$: $y_i = x_i \quad \forall i \in [1, n]$.
   b. לכל ניורון $Y_i$ (לפי סדר אקראי קבוע כלשהו):
      i. חשב את הקלט המעודכן שלו: $y_{in_i} = x_i + \sum_i y_j + w_{ji}$
      ii. בצע אקטיבציה: $y_i = \quad \mathbf{1}\ \boldsymbol{if}\ \boldsymbol{y_{in_i}} > \mathbf{0}\,, \quad y_i\ if\ y_{in_i} = 0,\ \ \mathbf{0}\ \boldsymbol{if}\ \boldsymbol{y_{in_i}} < \mathbf{0}$
      iii. שגר את הערך $y_i$ המעודכן לשאר הניורונים.
2. בדוק האם הוקטור התכנס לדפוס, אם כן עצור, אחרת חזור לשלב 1.

דוגמת הרצה: geeksforgeeks

משפט: האלגוריתם תמיד מתכנס ועוצר.

הוכחה: צ״ל שפונקציית האנרגיה תמיד יורדת ושהיא מתכנסת לגבול שלה.

**פונקציית האנרגיה:** $\boldsymbol{E = -\frac{1}{2}\sum_{i\neq j}\sum_j y_i y_j w_{ij} - \sum_i x_i y_i + \sum_j \theta_i y_i}$

אם האקטיבציה של הרשת משתנה ב$\Delta y_i$ אז האנרגיה משתנה ב: $\Delta E = -\left[\sum_j y_i w_{ij} + x_i - \theta_i\right]\Delta y_i$.

(קשר זה תלוי בעובדה שרק יחידה אחת יכולה לעדכן את הפעלתה בכל פעם).

נשקול את 2 המקרים בהם יופיע שינוי $\Delta y_i$ באקטיבציה של ניורון $Y_i$.

אם $y_i > 0$ הוא ישתנה ל0 אם $x_i + \sum y_j w_{ji} < \theta_i$. זה נותן שינוי שלילי עבור $y_i$, ולכן במקרה זה $\Delta E < 0$.

אם $y_i = 0$ הוא ישתנה למספר חיובי אם $x_i + \sum y_j w_{ji} > \theta_i$. זה נותן שינוי חיובי עבור $y_i$, ולכן גם במקרה זה $\Delta E < 0$.

לכן $\Delta y_i$ חיובי אם ורק אם $\sum y_j w_{ji} + x_i - \theta_i$ חיובי ו $\Delta y_i$ שלילי אם ורק אם אותה הכמות היא שלילית.

לכן, האנרגיה לא יכולה לגדול. לפיכך, מכיוון שהאנרגיה מוגבלת, הרשת חייבת להגיע לשיווי משקל יציב כך שהאנרגיה לא תשתנה במהלך איטרציות נוספות.

הערה: ניתוח פונקציית האנרגיה עבור רשת Hopfield מראה שהמאפיינים החשובים של הרשת המבטיחים התכנסות הם עדכון אסינכרוני של המשקולות, משקולות האפס באלכסון וסימטריות מטריצת המשקולות. בנוסף, זה לא חשוב אם אות חיצוני נשמר במהלך העיבוד או אם הכניסות וההפעלות הן בינאריות או ביפולריות.

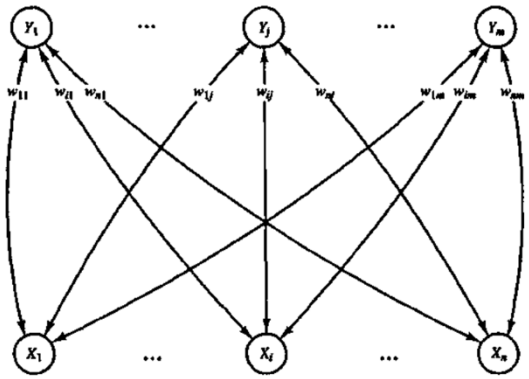
שימוש: קביעה האם וקטור נתון הוא ״מוכר״ או לא , ביצוע משימות שיוך אוטומטי ואופטימיזציה.

נכתב ע״י איתי לשובר ©

**BAM (Bidirectional associative memory) Network:**

הקדמה:

- זיכרון אסוציאטיבי דו-כיווני מאחסן קבוצה של אסוציאציות של דפוסים על-ידי סכימת מטריצות ביפולריות (מטריצת n על m עבור כל דפוס שיש לאחסן).
- הארכיטקטורה של הרשת מורכבת משתי שכבות של נוירונים, המחוברות בנתיבי חיבור דו כיווניים.
- הרשת איטרטבית, כלומר, שולחת אותות הלוך ושוב בין שתי השכבות עד שכל הנוירונים מגיעים לשיווי משקל (עד שהפעלת כל נוירון נשארת קבועה למשך מספר שלבים).
- כיוון שהמשקולות הן דו-כיווניות והאלגוריתם עובר לסירוגין בין עדכון הפעלות עבור כל שכבה, נתייחס לשכבות כשכבת ה-X ושכבת ה-Y (ולא לשכבות הקלט והפלט).




ארכיטקטורה:

לרשת יש $n$ יחידות בשכבת ה-$X$ שלה ו-$m$ יחידות בשכבת ה-$Y$ שלה. החיבורים בין השכבות הם דו-כיווניים, כלומר, אם מטריצת המשקל לאותות הנשלחים משכבת ה-$X$ לשכבת ה-$Y$ היא $W$, מטריצת המשקל עבור אותות הנשלחים משכבת ה-$Y$ לשכבת ה-$X$ היא $W^T$.

אלגוריתם:

שתי הצורות הדו-ערכיות (בינאריות או ביפולריות, הוכח כי וקטורים ביפולרים משפרים את ביצועי הרשת) של BAM קשורות קשר הדוק. בכל אחד מהם, המשקולות נוצרים מסכום התוצרים החיצוניים של הצורה הביפולרית של זוגות הווקטורים באימון.

כמו כן, פונקציית האקטיבציה היא פונקציית צעד (treshold function12), עם אפשרות לtreshold שאינו אפס.

0. אתחול המשקולות המאוכסנות – קבוצה של P זוגות $s, t$ של וקטורים: $w_{ij} = \sum_{s_i, t_i \in P} s_i \cdot t_i^T$
1. לכל זוג קלטים בצע:
   a. אתחל את נוירוני שכבת $X$ להיות הערכים של וקטור הקלט עם הגודל המתאים $(n)$.
   b. אתחל את נוירוני שכבת $Y$ להיות הערכים של וקטור הקלט עם הגודל המתאים $(m)$.
   c. כל עוד לא הגענו לתנאי העצירה:
      i. עדכן את הערכים בשכת הY:
         חשב את הקלט לכל $Y_j$: $y_{in_j} = \sum_i w_{ij} x_i$.
         הפעל את פונקציית האקטיבציה: $y_j = f(y_{in_j})$ (*threshold function*).
         שלח סיגנל (את הפלט) לשכבת הX.
      ii. עדכן את הערכים בשכת הX:
         חשב את הקלט לכל $X_j$: $X_{in_i} = \sum_i w_{ij} y_j$.
         הפל את פונקציית האקטיבציה: $x_i = f(X_{in_j})$ (*threshold function*).
         שלח סיגנל לשכבת הY.
      iii. בדוק האם יש התכנסות.
         אם וקטורי ההפעלה x ו-y הגיעו לשיווי משקל, אז עצור, אחרת, חזור לשלב c.

הערה: כל אחת מוקטורי הקלט עשוי להיות וקטור האפס.

דוגמת הרצה: geeksforgeeks

משפט: האלגוריתם מסתיים.

הוכחה: ניתן להוכיח את ההתכנסות של רשת BAM באמצעות פונקציית אנרגיה (yapunovl). פונקציית ליאפונוב חייבת להיות יורדת וחסומה. עבור רשת BAM, פונקציה מתאימה היא הממוצע של אנרגיית האות עבור מעבר קדימה ואחורה:

**פונקציית האנרגיה:** $\boldsymbol{L = -\frac{1}{2}(xWy^T + yW^Tx^T)}$

מכיוון ש$xWy^T$ ו$yW^Tx^T$ הם סקלרים, והtransopse של סקלר הוא סקלר, ניתן לפשט את הביטוי הקודם:

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

עבור פונקציות צעד בינאריות או ביפולרית, פונקציית Lyapunov חסומה מלמטה בבירור על ידי: $\boldsymbol{-\sum_{j=1}^{m}\sum_{i=1}^{n} |w_{ij}|}$.

שימוש: שיוך אותיות לקודים ביפולרים פשוטים. זיכרון מוגבל $\min(n, m)$.

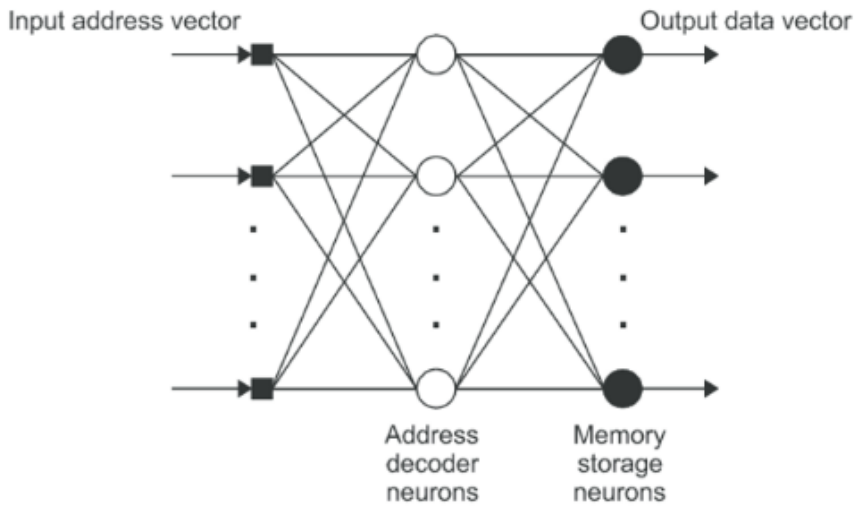
נכתב ע"י איתי לשובר ©

**SDM:**

הקדמה:

- ניתן להתייחס ל-SDM כאל הרחבה של זיכרון גישה אקראית (RAM) או כסוג מיוחד של רשת עצבית מזרימה שלוש שכבות.
- בהינתן מרחב מידע גדול מדי, שלא ניתן להחזיק את כולו, נרצה להחזיק $m$ נקודות במרחב שיקראו hard location שבעזרתם נוכל לתאר ולקבל את מרחב המידע כולו (בדומה לזיכרון RAM).
- ניתן להשתמש במודל זה על מנת להחזיק כמות כתובות גדולה יותר ממה שהמחשב מסוגל להכיל.

ארכיטקטורה:




אלגוריתם:

1. בהינתן כתובת וירטואלית נמצא את כל הכתובות האמיתיות שהן במרחק hamming מסוים מכתובת זו.
2. עבור כל הכתובות במרחק המתאים נחבר את וקטורי המידע שהן מחזיקות לכדי וקטור אחד.
3. בוקטור 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.
4. הכתובת הוירטואלית נכנסת כקלט לשכבה הראשונה, לכל נוירון בשכבה השנייה המשקולות הנכנסות אליה מייצגות את כתובתו בזיכרון האמיתי.
5. המשקלות שיוצאות מהנוירון בשכבה האמצעית מייצגות את המידע שהוא מצביע עליו.

| Ref. inp. data | Radius | Dist. | Select | Associated input data | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0101101110 | 2 | | | | | | | | | | | | | |
| 0001001010 | | 3 | 0 | | | | | | | | | | | |
| 0000011100 | | 5 | 0 | | | | | | | | | | | |
| 0001101100 | | 2 | 1 | 1 | -2 | -5 | 3 | 0 | -4 | 3 | 1 | 4 | -3 | |
| 0110111010 | | 4 | 0 | | | | | | | | | | | |
| 1000110100 | | 6 | 0 | | | | | | | | | | | |
| 0000000100 | | 5 | 0 | | | | | | | | | | | |
| 0111101110 | | 1 | 1 | 2 | -1 | -1 | 0 | 2 | -2 | -1 | 3 | -1 | -1 | Counters |
| 0011001010 | | 4 | 0 | | | | | | | | | | | |
| 0000011100 | | 5 | 0 | | | | | | | | | | | |
| 0101101010 | | 1 | 1 | 1 | -1 | -2 | 1 | 0 | -1 | 2 | 1 | 3 | -2 | |
| 0000110110 | | 4 | 0 | | | | | | | | | | | |
| 0010011110 | | 5 | 0 | | | | | | | | | | | |
| Location addr. | | | | | | | | | | | | | | |
| | | | Sum | 4 | -4 | -8 | 4 | 2 | -7 | 4 | 5 | 6 | -6 | |
| | | | Thresholding | | | | | | | | | | | |
| | | | Output data | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | |

Hebb rule:

כאשר שני ניורונים המקושרים אחד לשני באופן דו כיווני פועלים במקביל, נרצה הקשר ביניהם יתחזק: $w_{ij} = \frac{1}{p}\sum_{k=1}^{p} x_i^k x_j^k$

כאשר $p$ הוא מספר הניורונים ברשת ו$x_i^k$ הוא הקלט ה$k$ בניורון ה$i$.

למעשה, זהו חוק שלפיו מאתחלים את המשקולות בכל האלגורתימים עם זיכרון אסוציאטיבי (SDM, BAM, Hopfield).

שימוש: Cerebellum Model, Bit-Map transformation (Hebrew-English translation).

נכתב ע״י איתי לשובר ©

**מושגים נוספים:**

Crosstalk: הפרעה המתרחשת כאשר הדפוסים המאוחסנים בזיכרון אסוציאטיבי אינם אורתוגונליים הדדיים.
אם ניתנת לרשת אחת מהדפוסים המאוחסנים כקלט, התגובה תהיה שילוב של הפלט הרצוי ושל דפוסי היעד עבור הדפוסים המאוחסנים האחרים שאינם אורתוגונליים לדפוס הקלט.

Conscience: מנגנון המונע מכל cluster שנוצר ברשת counter propogation לטעון לחלוקה לא הוגנת של וקטורי הקלט, סביר להניח שהclusterים הראשונים יהיו בעלי יתרון.
מנגנון מצפון משמש לשיפור יעילות הלמידה ברשתות עצביות מלאכותיות. מנגנון זה מאפשר לבטל את ההשפעה של מה שמכונה "נוירונים מתים", שאינם לוקחים חלק בתחרות בשלב הלמידה. לנוירונים אלה יש בדרך כלל השפעה מזיקה על ביצועי הרשת.

Momentum: שינוי נפוץ לאימון בBackPropogation.
בכל שלב, תיקון המשקולות מתבסס על שילוב של תיקון המשקל הנוכחי ושינוי המשקל מהשלב הקודם.
למעשה, כדי להימנע ממינומים מקומי, אנו משתמשים במונח מומנטום בפונקציית המטרה, שהוא ערך בין 0 ל-1 שמגדיל את גודל הצעדים הנקטים לעבר המינימום על ידי ניסיון לקפוץ מחוץ למינימום מקומי.
אם טווח המומנטום גדול אז יש לשמור על קצב הלמידה ($\alpha$) קטן יותר.
ערך גדול של מומנטום אומר שההתכנסות תתרחש מהר, אבל אם המומנטום וגם קצב הלמידה נשמרים בערכים גבוהים, אז אנו עלולים לדלג על המינימום בצעד גדול מדי.
מצד שני, ערך קטן מדי של מומנטום אינו יכול להימנע באופן מהימן ממינימום מקומי, ויכול גם להאט את האימון של המערכת.

Equiprobable: אחרי שמאמנים רשת Kohonen, מדד לאיכות הפריסה שלה היא שהיא תהיה equipropaple.
התכונה הזו אומרת שבהינתן דוגמה חדשה מאותה התפלגות שעליה התאמנו- הסיכוי שהיא תנחת בכל אחד מהclusterים שווה. במילים אחרות- כל cluster אחראי על אותו מספר של דוגמאות.

Fault tolerant retrieval: סובלנות תקלות מתייחסת ליכולתה של רשת להמשיך לפעול ללא הפרעה כאשר אחד או יותר מהרכיבים שלה נכשלים.
המטרה של יצירת מערכת סובלנית לתקלות היא למנוע שיבושים הנובעים מנקודת כשל בודדת, הבטחת זמינות גבוהה והמשכיות עסקית של יישומים או מערכות קריטיות למשימה.

| | | | |
|---|---|---|---|
| **Perceptron** | Classification | Shape recognition | |
| **Adaline** | Classification | Shape recognition | |
| **MP** | Classification | | Representation of some Boolean functions |
| **Feed Forward** | Classification | Shape recognition | Representation of some Boolean functions |
| **Kohonen** | Clustering | Self-Organizing Map | unsupervised learning |
| **Hopfield** | Associative Memory | Fault tolerant Memory | unsupervised learning |
| **Counter Propogation** | Clustering | | unsupervised learning |
| **Hamminng** | Clustering | | unsupervised learning (???) |
| **MaxNet** | Competitive | | |
| **SDM** | Associative Memory | Fault tolerant Memory | |
| **BAM** | Associative Memory | Fault tolerant Memory | |

נכתב ע״י איתי לשובר ©