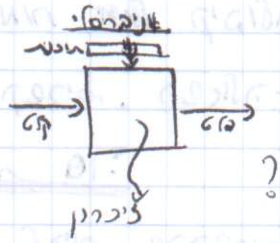
16/5/08

נוירו-חישובים

שקופית 3:



(*) התוכנית הינה זו שקובעת מה יהיה הפלט.

(*) מה היתרון לתוכנית שבתוך ה-CPU (פקודות בזיכרון)?

שינוי תוכן כפי רוצה -

שקופית 4:

(*) מה ההבדל בין מה שהמוח עושה לבין מה שתוכנית עושה?

כשעושים טעות בתוכנית ניתן למצוא בקומפילציה אולם כשהמוח מחשב "התוכנית" אינה "שבירה", fault tolerant. האלמנט של המוח שמחשב שונה מהאלמנט של תוכנית שמחשבת כיוון שהמוח יכול ללמוד מדוגמאות.

דוגמא- הכנסת סטודנט לכיתה רק אם הוא כבר שילם שכ"ל.

(*) שמים מצלמה עם מנעול על הדלת כאשר סטודנט רוצה להיכנס המצלמה מצלמת את הסטודנט נעשה סריקה על פני הסטודנט על התכונה שלו האם שילם ואם כן הדלת נפתחת..

תוכנית תמיד תבצע את הסריקה הנ"ל אולם מוח יכול לבצע הכללה ולאחר מס' פעמים כבר לומד ואינו מבצע את הסריקה הנ"ל.

שקופית 5:

(*) המוח עובד לפי מיליארדים של אלמנטים (דם, נוירונים, ביוכימיה וכו' ...)

(*) אנו נסתכל על עצבים - כמה זמן לוקח לעצב לעבוד: 10^{-3}

מחשבים עובדים בסדר גודל יותר מהיר: 10^{-9} (צ'יפים)

(*) אולם המוח עובד יותר ביעילות

שקופית 6:

(*) המוח יותר טוב מבחינת חושים, הכללות, fault tolerant

שקופית 7:

(*) בינה מלאכותית מקביל לפסיכולוגיה קוגניטיבית - תכונות של המוח

חישובים נוירונים מקביל נוירופסיכולוגיה

שקופית 9:

(*) במוח אין רק נוירונים אלא גם קשרים ביניהם (בערך 10^{16} קשרים)

⇐ למוח ~~...~~ קיבולת יכולה להיות גדולה יותר ממחשב אם מסתכלים על הקשרים. השאלה - מה יושב על הקשרים?

שקופית 10:

(*) יכול להיות שהקשרים אינם משמעותיים

(*) אם נסתכל על סרטן אין לו הרבה עצבים והוא מבצע פעולות כגון: מציאת אוכל, מציאת בית וכו' ולכן יכול להיות שלא צריך כל-כך הרבה עצבים.

שקופית 11:

משנות ה-50, לעצבים יש תאום נפרדים (לא "יחוד") בנוסף לזה יש Spikes (אות חשמלי) שאו שיש אותם או שאין (0 או 1) - שולח את ה-Spike או לא שולח אותו.

axon של עצב מקבל dendries של עצבים אחרים.

הם ביצעו הפשטה ל- dendries שהם וקטור של אינפורמציה וגם ישנם משקלים שמגיעים לתא והתא מבצע קומבינציה לינארית: $\sum_{i=1}^{n} w_i x_i$ ואז יוצאים Spikes אם זה עובר איזשהו סף.

שקופית 12:

בעצם זה מגדיר פונקציה בוליאנית אבל השאלה היא איזה פונ' בוליאנית?

אם משנים את ה-w אזי נקבל פונ' בוליאנית אחרת (גם את הסף ניתן לשנות)

(*) נשים לב שהאלמנט של זמן אינו מופיע במודל

שקופית 13:

ניתן להציג כל פונ' בוליאנית, כיוון שניתן להציגה בצורה נורמטיבית

המשקלים הם שערים - and, or, not.

and - איחוד של שערים: 1 סף: 1.5 ($n-\frac{1}{2}$ ערכים)

NOT - שערים: -1 סף: 0.5

CNF

x_1	x_2	x_3	...	x_n	$f(x_1,...,x_n)$
T	T			T	T
F	T			T	F
T	F			T	⋮
F	F			T	
⋮	⋮			⋮	
				F	

2^n שורות

כל ערך פה מגדיר פונ' שונה

$x_1 \wedge x_2 \wedge ... x_n$
T
F
F
⋮

הכל F

$\neg x_1 \wedge x_2 \wedge ... x_n$
F
T
T
⋮

הכל T חוץ מ-x_1

⇐ כל פונ' ניתן להציג מ-$\wedge$ ו-$\neg$.

שקופית 20:

לרשת נוירונים יש כוח אולם כיצד ניתן להשתמש בה?

אנו רוצים רשת פשוטה שיהיה קל לתפעל, כלומר רשת בעלת חוקים למצי

(כללים) יוניפורמיים לקביעת המשקלים

שקופית 21:

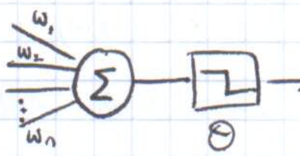
(*) רשת עם נוירון אחד: ⇐ האם ניתן לייצג כל פונ' בוליאנית ע"י נוירון אחד?

(*) השאלות שמעניינות: ⇐ מה הכוח של נוירון אחד?

⇐ האם יש שיטה אוטומטית ללימוד פונ' ע"י דוגמאות?

	כח (מה ניתן להציג)	שיטת לימוד (אלגוריתם)
3-level NNs	כל פונ' בוליאנית	?
נוירון אחד	?	? ← נענה היום. (perceptron)

שקופית 22:



שיטה אפקטיבית למציאת המשקלים:

$\vec{w}, \theta$

$\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$ (אוסף דוגמאות)

לכל $\vec{x} \in \mathcal{F}^+$ (דוגמאות חיוביות) | $\vec{x} \in \mathcal{F}^-$ (דוגמאות שליליות)

$\left(\sum_{i=1}^{n} w_i x_i =\right) \vec{w}\cdot\vec{x} > \theta$ | $\vec{w}\cdot\vec{x} < \theta$

<u>לימוד</u> - אם נכון החזר, (קח דוגמא נוספת) אחרת תקן לפי דוגמא.

התכנסות - כאשר ניקח את כל הדוגמאות ולא נצטרך לבצע תיקון לאף אחד.

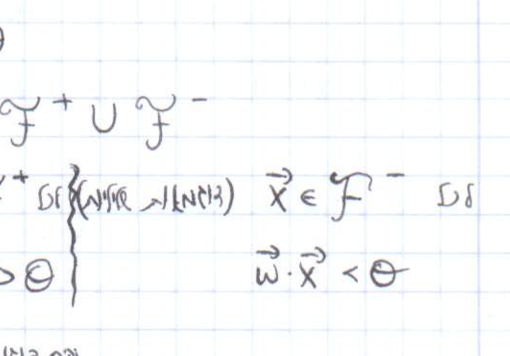
שקופית 23:

<u>בה"כ:</u> $\theta = 0$ נצרף את ה-x במימד אחד יותר גדול. (שערכו יהא 1)

$\sum w_i x_i > \theta \iff \sum_{i=1}^{n} w_i x_i - \theta > 0 \iff \sum_{i=1}^{n+1} w_i x_i > 0$

$\vec{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \longrightarrow \hat{\vec{X}} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$

$\sum_{i=1}^{n} w_i x_i + \underset{-\theta}{w_{n+1}} \cdot \underset{1}{x_{n+1}} > 0$



		x_1	x_2	And	x_3
$\mathcal{F}^+$	$\vec{x_1}$	1	1	1	1
$\mathcal{F}^-$	$\vec{x_2}$	1	0	0	1
	$\vec{x_3}$	0	1	0	1
	$\vec{x_4}$	0	0	0	1

⇐ לא משנה התוצאה תישאר זהה.

קיים $\vec{V}$ כך ש:

$\vec{V}\cdot\vec{X} > 0 \quad x\in \mathcal{F}^+$

$\vec{V}\cdot\vec{X} < 0 \quad x\in \mathcal{F}^-$

צריך למצוא $\vec{w}$ כזה, כיצד?

<u>בה"כ:</u> $\mathcal{F} = \mathcal{F}^+$

$x \in \mathcal{F}^- \Leftrightarrow -x \in \mathcal{F}^+$

$\sum w_i x_i < 0 \Leftrightarrow \sum w_i(-x_i) > 0$

<u>בה"כ:</u> ניתן לנרמל את הוקטורים

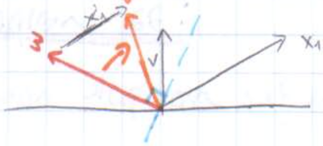
נחפש $\vec{w}\cdot\vec{x} > 0$ לכל $\vec{x}\in \mathcal{F}^+$

כלומר, בה"כ נניח לכל $\vec{x}\in \mathcal{F}^+$ $\|\vec{x}\|=1$ נמצא $\vec{w}$ כך ש: $w\cdot\frac{\vec{x}}{\|x\|}>0$

(*) אנו רוצים למצוא אלגוריתם שיכנס בזמן סופי לפיתרון
המשפט אומר שקיים כזה. (שקופית 26)

$\vec{w}\cdot\vec{x}_1 \not> 0$ מכיוון שהזוית גדולה מ-90°

לכן נבצע $w+x$



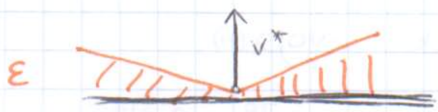
אולם יכול להיות שה-w החדש מקלקל לדוגמא
אחרת ולכן יש שוב לבצע בדיקה עד שמגיעים ל- $w=v$ (כלומר, w שכל דוגמא יתן
פיתרון נכון).

<u>צ"ל:</u> שאכן מגיעים להתכנסות הנ"ל $(w=v)$

<u>שקופית 30:</u>

נוסיף שקיים v^* בעל תחום הרבה ε (המינימום של $\mathcal{F}$)

$v^*\cdot x > \varepsilon > 0$ לכל $\vec{x}\in \mathcal{F}$



כלומר, הולכים ל-Fix רק מס' סופי של
פעמים.

<u>שקופית 35:</u>

בה"כ $\|v^*\|=1$ אם דו-מימדי אז זהו: $\cos \le 1$ $\rightarrow \frac{V^*\cdot w}{\|w\|}\le 1$

<u>הערה:</u> ניתן לבחור $w = w_0$

נראה מה קורה לחלק עליון ומה קורה לחלק תחתון ונוכיח שאי-אפשר להחליף
יותר ממס' קבוע של פעמים.

$\vec{w}_{n+1}=\vec{w}_n+x$

(I) $v^*\cdot w_{n+1} = v^*\cdot w_n + v^*\cdot x > v^*\cdot w_n+\varepsilon$

$\Rightarrow v^*\cdot w_n \ge n\varepsilon$ (n פעם שהולכים ל-Fix ←)

(II) $\|w_{n+1}\|^2 = (w_{n+1})(w_{n+1}) = (w_n+x)(w_n+x)$

$= w_n w_n + 2w_n x + x\cdot x$

(II) $\|w_{n+1}\|^2 = (w_{n+1})(w_{n+1}) = (w_n+X)(w_n+X)$

$= w_n w_n + 2w_n X + X\cdot X = \|w_n\|^2 + \underbrace{\|X\|^2}_{=1} + \underbrace{2\vec{w}_n\vec{X}}_{\text{הלכנו ל-fix, לכן } <0}$

$< \|w_n\|^2 + 1$

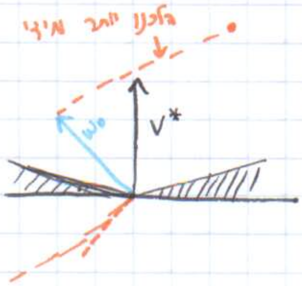
$\Rightarrow \|w_n\|^2 < n \quad \Rightarrow \|w_{n+1}\|^2 < \|w_0\|^2 + n \longrightarrow$ תלוי מה נבחר w_0

$\Rightarrow \|w_n\| < \sqrt{n}$

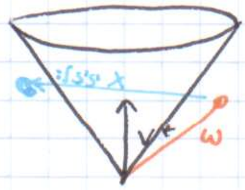
כלומר המונה גדול מ-εn והמכנה קטן מ-$n+1$.

$1 \geq \frac{V^* \cdot w_n}{\|w\|} \geq \frac{n\varepsilon}{\sqrt{n}} \quad \Rightarrow \quad 1 \geq \frac{n\varepsilon}{\sqrt{n}} = \sqrt{n}\varepsilon$

$\Rightarrow \boxed{\frac{1}{\varepsilon^2} \geq n}$



ולכן נלך ל-fix מס' סופי של פעמים.



$\Rightarrow$ כל הוקטורים בתוך הקונוס הוא טוב מספיק לספק את x.

(*) הוקטור w לא היה מספיק טוב וכשעשינו תיקון ~~[illegible]~~ עברנו גם למקום שאינו מספיק טוב.

(*) לתרגל את האלגוריתם על AND.

23/5/08

נירו-חישוביים

(*) שתכנן את הנירון זה לבחור את המשקלים (והסף)

שקופית 31:

(*) דוגמא עבור AND:

X_1	X_2	X_3	And
0	0	1	0
0	1	1	0
1	0	1	0
1	1	$\boxed{1}$	1

← תשובה רצויה

לא משנה את התוצאה

(*) תוספת ה[illegible] אינה משפיעה על התוצאה

(*) באלגוריתם perceptron נותנים משקל לכל x_i
אם בסוף התוצאה לא נותנת תוצאה רצויה
משנים את המשקל.
אם משנים כל פעם w אחד: one in a time
אם משנים עבור כל ה-w: Batch

(*) דוגמא נוספת:

	X_1	X_2	X_3	And
(1)	0	0	1	0
(2)	1	0	1	0
(3)	0	1	1	0
(4)	1	1	1	$\boxed{0}$
(5)	0	0	1	$\boxed{1}$
(6)	1	0	1	$\boxed{1}$
(7)	0	1	1	0
(8)	1	1	1	$\boxed{0}$
(9)	0	0	1	0
(10)	1	0	1	$\boxed{1} \to \vec{w} = \begin{pmatrix}0\\2\\-1\end{pmatrix}$
(11)	0	1	1	$\boxed{1} \to \vec{w} = \begin{pmatrix}0\\1\\-2\end{pmatrix}$
(12)	1	1	1	$\boxed{0} \to \vec{w} = \begin{pmatrix}1\\2\\-1\end{pmatrix}$
	⋮			

$\Rightarrow \quad w_1^0 X_1 + w_2^0 X_2 + w_3^0 X_3 = 0$

כאשר $w_i^0 = 0$

← לא נכון ← $\vec{w} = w^0 + X \Rightarrow w^{new} = \begin{pmatrix}0\\0\\0\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}1\\1\\1\end{pmatrix} \Rightarrow w_i = 1$

$\Rightarrow \quad w_1^{new} X_1 + w_2^{new} X_2 + w_3^{new} X_3 =$

(5) $1\cdot 0 + 1\cdot 0 + 1\cdot 1 = \boxed{1} \rightarrow$ לא נכון

$w^{new} = w - X = \begin{pmatrix}1\\1\\1\end{pmatrix} - \begin{pmatrix}0\\0\\1\end{pmatrix} = \begin{pmatrix}1\\1\\0\end{pmatrix}$

(6) $1\cdot 1 + 1\cdot 0 + 0\cdot 1 = \boxed{1} \rightarrow$ לא נכון

$\vec{w} = w - X = \begin{pmatrix}1\\1\\0\end{pmatrix} - \begin{pmatrix}1\\0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\-1\end{pmatrix}$

(7) $0\cdot 0 + 1\cdot 1 + (-1)\cdot 1 = 0$

(8) $0\cdot 1 + 1\cdot 1 + (-1)\cdot 1 = \boxed{0} \rightarrow$ לא נכון

$\vec{w} = w + X = \begin{pmatrix}0\\1\\-1\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}1\\2\\0\end{pmatrix}$

תיקון:

(אם היה צריך להיות 1) $\vec{w}^{new} = \vec{w}^{old} + X$

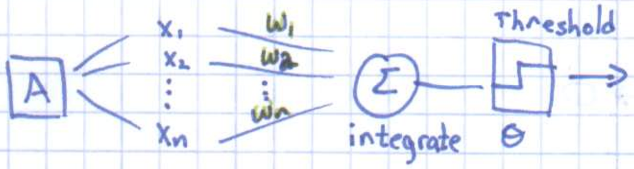
(אם היה צריך להיות 0) $\vec{w}^{new} = \vec{w}^{old} - X$

וממשיכים עד שהכל נותן תוצאות נכונות ואלו הם המשקלים

אם יש משהו שניתן להפרדה לינארית האלגוריתם מתכנס.

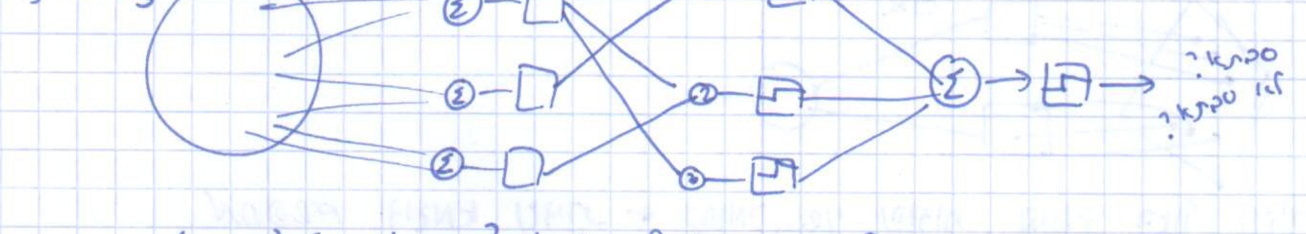
(*) דוגמא:

בחירת משקלים משפיעה אם הנירון להיות not, or, and



האם זה A או לא

נניח שאני רוצים לבנות מזהה תמונות (האם סבתא בתמונה)

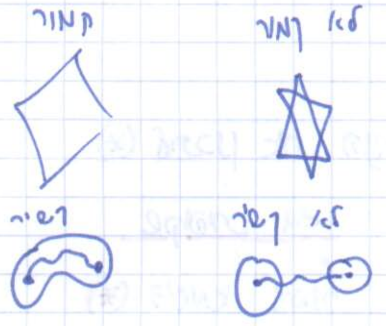


השאלה: כמה features ו- higher features צריך ואיך למצוא את ה- w_i ?

שאלה:

רוצים לראות האם הצורה קמורה

- " - קשירה



ניקח מס' גופים ונבדוק האם הם קשירים או לא.

$\sum w_i x_i \le 0$ $\quad \sum w_i x_1 > 0$ $\quad \sum w_i x_i < 0$ $\quad \sum w_i x_i > 0$ $\Leftarrow$ פ גופים קשירים ו-2 לא

$\downarrow$

$$\sum_L w_i x_i + \sum_M w_i x_i + \sum_R w_i x_i \le 0$$

$$LHS^1 + RHS^1 < C \qquad LHS^2 + RHC^2 > C \qquad LHS^3 + RHS^3 < C \qquad LHS^4 + RHS^4 > C$$

$$\left.\begin{array}{l} LHS^1 = LHS^2 \\ RHS^1 = RHS^4 \\ RHS^2 = RHS^3 \\ LHS^3 = LHS^4 \end{array}\right\} \Leftarrow$$

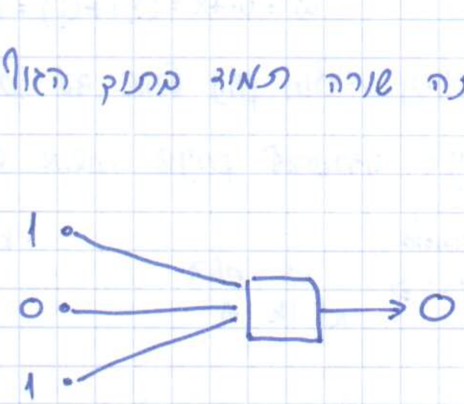
יש סתירה ולכן לא ניתן לפתור

כלומר, אין perceptron שפותר את הבעיה.

נבדוק קמירות.

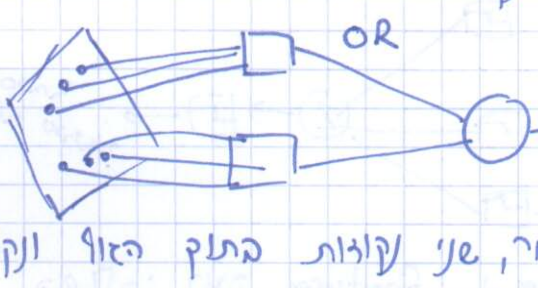
להיות קמור = 3 נקודות באותה שורה תמיד בתוך הגוף.

נבנה מכונה בצורה הבאה:



(*) רק עבור הקומבינציה הזאת נקבל 0 אחרת 1.

נשים סנסור כזה על כל 3 נקודות:



מחפשים דוגמא נגדית ← כלומר, שני נקודות בתוך הגוף ונקודה אמצעית לא בתוך הגוף.

יש מכשיר שאנו רוצים לדעת מה הוא עושה

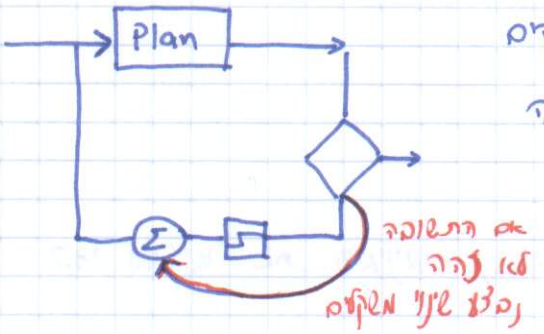
אנו לוקחים את ה-input ובודקים

האם המכשיר שלנו מוצא את אותה

התוצאה כמו ה-plan

אם לא אז חוזרים ומשנים

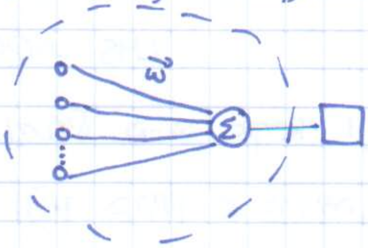
את ערכי ה-w



Adaline

אלגוריתם נוסף:

אם לוקחים פונ' לינארית שהיא פונ' של פונקציות לינאריות ונסתכל בחלק של נירון:



נרשום את הרשימה:

$< \vec{x}^{(1)} d^{(1)} >$ ← התשובה הרצויה

$< \vec{x}^{(2)} d^{(2)} >$

$\vdots$

$< \vec{x}^{(m)} d^{(m)} >$

יש לנו את ה-$\vec{w}$ ההתחלתי ונראה כמה קומבינציות לינאריות נותנות את התוצאה הרצויה.

$$E(\vec{w}) = \sum_{j=1}^{n} \left(\Sigma w_i x_i^{(j)} - d^{(j)} \right)^2$$ ← הטעות

רוצים למצוא את המינימום (ע"י גזירה)

* מכיוון שזה בריבוע אז זוהי פרבולה ולכן יש מינימום אחד

דוגמא עבור AND:

x_1	x_2	x_3	And
0	0	1	-1
1	0	1	-1
0	1	1	-1
1	1	1	1

$$E(w) = (w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot 1 - (-1))^2 + (w_1 \cdot 1 + w_2 \cdot 0 + w_3 \cdot 1 - (-1))^2 +$$
$$(w_1 \cdot 0 + w_2 \cdot 1 + w_3 \cdot 1 - (-1))^2 + (w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 - 1)^2 =$$

נגזור כל חלק בנפרד:

$$\frac{\partial E}{\partial w_i} = 2(w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot 1 + 1) + 2(\ldots) + 2(\ldots) + 2(\ldots)$$

$$\frac{\partial E}{\partial w_1} = \sum_{j=1}^{n} 2\left(\sum_{i=1}^{n} w_i x_i^{(i)} - d^{(j)}\right) X_1$$

$$\frac{\partial E}{\partial w_2} = \sum_{j=1}^{n} 2\left(\sum_{i=1}^{n} w_i x_i^{(j)} - d^{(i)}\right) X_2$$

$$\vdots$$

כדי למצוא את המינימום נשווה הכל ל-0 ונקבל n שוויוניות ב-n נעלמים

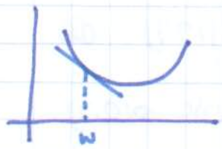
$$\frac{\partial E}{\partial w_1} = 0 \quad \dots \quad \frac{\partial E}{\partial w_n} = 0$$

ניתן לרשום את זה כך: $M\vec{w} = \vec{q}$

$\vec{w} = M^{-1}\vec{q}$ ← קשה למצוא את M^{-1} (עולה הרבה פעולות מחשב)

איך ניתן לפתור בלי להפוך מטריצה?

שיטת LMS



נתחיל ב-w אקראי ונבדוק את הגרדיינט (משיק)

נלך בכיוון של הירידה, כלומר, נשנה את ה- w_i

בקצת ונבדוק שנית.

$$\vec{w}^{new} = \vec{w}^{old} - \alpha \vec{\nabla}_w E(\vec{w})$$

גרדיינט

$\alpha =$ גודל הצעד

6/6/08

נירו-חישוביים

(*) עבור XOR ב-Adeline היתה התכנסות לעומת perceptron ~~...~~ שהיה יכול לעבוד עד אינסוף.

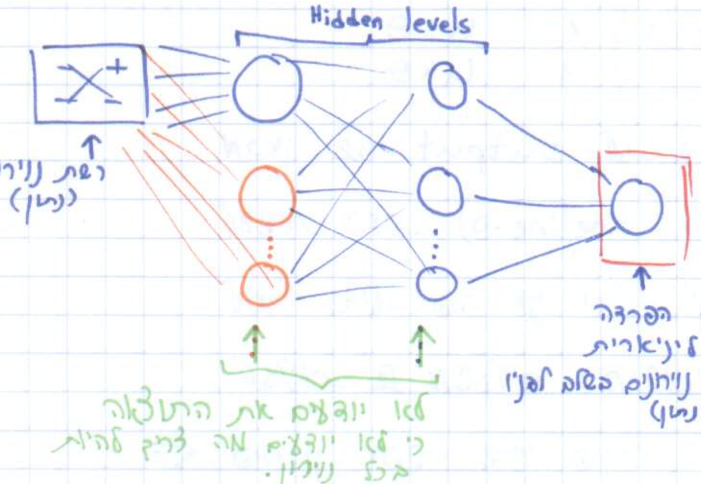
(*) בתרגיל נבדק:

- התכנסות

- Test set vs. Training Set

נשים לב - הסתכלנו על דברים אשר ניתנים להפרדה לינארית

ואנו מחפשים אלג' אשר מסתכלים על רשת של נוירונים (ולא נוירון אחד)



* מה אם יש לנו כמה?

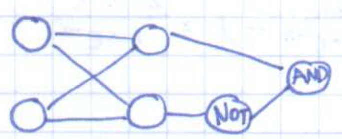
* בעצם ניתן להסתכל על כך כביצוע טרנספורמציה על הנתונים.

זהו בעצם feed forword.

השאלה - איך ייתן לנו תוצאה של הכל? נשים לב שבסוף אנו יודעים את התוצאה

הארכיטקטורה נתונה ואנו מחפשים את המשקלים.

* כיצד מוצאים את המשקלים הנ"ל?



בעצם ניתן לתרגם ל:

נשים לב שב-perceptron לא ניתן לבצע הכללה לכמה רמות אך ב-Adeline כן.

נמצא אלגוריתם אשר ייתן לנו את המשקלים:

יש לנו Training Set $\langle \vec{x}, \vec{z} \rangle$ ⇐ אוסף דוגמאות

(*) $\vec{z}$ כיוון שיכול להיות שיהיו אוסף אפשרויות

ספרה (1/0) (בינארי)

אם יש d מימדים

$$\text{Error}(\vec{w}_{\text{משקלים}}) = \frac{1}{2} \sum_{k \in \text{training examples}} \sum^{d} (t_k^d - y_k^d)^2$$ ← הטעות ברשת

t_k = training Example

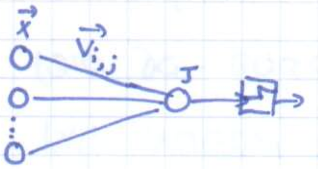
y_k = Net Output $(\vec{x}_k)$

יש מס' בעיות:

- ב-Adeline היה פשוט כי היה נוירון אחד ולכן משקל אחד. כאן ישנם מס' נוירונים ולכן כאן לא ברור מהיכן מגיעה הטעות.

⇐ נמצא את המשקל של E, כלומר, נמצא נגזרת לפי כל ה-w ולקחת את הוקטורים ולמצוא את הכיוון ללכת על-פיו.

$$\underset{\text{משקלים}}{\nabla} E(\ ) \rightarrow \left(\frac{\partial E}{\partial w}, \dots\right)$$



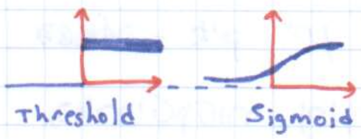
- אם נסתכל על:

$$Output_J = f\left(\sum x_i v_{ij}\right)$$

$$f(k) = \begin{cases} 1 & u \geq \theta \\ -1 & u < \theta \end{cases} \leftarrow \text{לא ניתנת לגזירה}$$

מכאן שה-output לא ניתנת לגזירה ולכן אנו לא יכולים לעשות זאת (ב-Adeline הסתכלנו על "חלק מהסכום" בלבד ואח"כ בדקנו לסף ואילו פה אנו יכולים לעשות זאת רק בסוף ← לא נוכל למצוא טעויות של משקלים בחישובי ביניים).

⇐ פיתרון אפשרי שינוי פונקציית הסף לפונ' שכן ניתנת לגזירה



לדוג':

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

בעצם:

$$\lim_{x\to\infty}\left(\frac{1}{1+e^{-x}}\right) = 1$$

Approximate for ← Binary

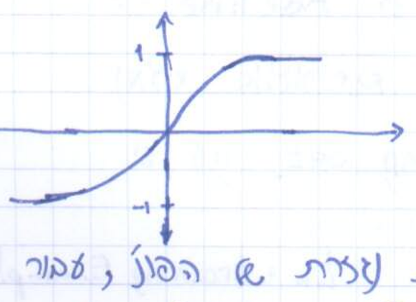
$$\lim_{x\to-\infty}\left(\frac{1}{1+e^{-x}}\right) = 0$$

for Bipolar ← $g(x) = 2f(x) - 1 = \frac{2}{1+e^{-x}} - 1 = \frac{2-1-e^{-x}}{1+e^{-x}} = \frac{1-e^{-x}}{1+e^{-x}}$

↑ Bipolar Sigmoid arctanh (hyperbolic tanges)

$$\lim_{x\to\infty}\left(\frac{2}{1+e^{-x}}-1\right) = 1$$

$$\lim_{x\to-\infty}(g(x)) = -1$$



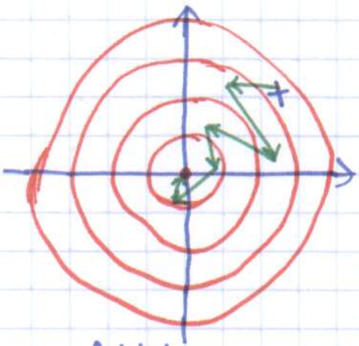
$$h(x) = \arctan x$$

* מכיוון שאנו לוקחים נגזרת נצטרך לקחת נגזרת של הפונ', עבור $f(x)$ ניקח:

$$f'(x) = \frac{1}{(1+e^{-x})^2}(e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right) = f(x)(1-f(x))$$

אלגוריתם Back Propagation

* ב- Adeline ביצענו $Error(\vec{w}) = \frac{1}{2}\sum_{dimension}(t_k^d - y_k^d)^2$



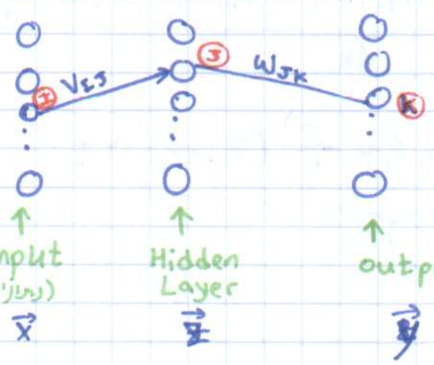
x - המיקום שהיינו

← - ביצוע תיקון

• - נקודה רצויה

התיקון התבצע ב- batch

* ב-Back Propagation נבצע "אותו דבר" רק שאת התיקון נבצע One at a time.



$\vec{z}$ - המצב של הנוירונים באמצע.

$$y_k = \underset{\text{הפונק' שבחרנו}}{f}\Big(\sum \underbrace{w_{Jk} z_J}_{\text{מה נכנס ל-y}}\Big)$$

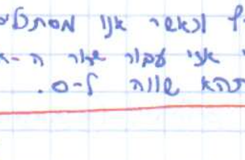
$$z_J = f\Big(\sum \underbrace{V_{IJ} X_I}_{\text{מה נכנס ל-J}}\Big)$$

* הסף- נוסיף נוירון שתמיד 1 והמשקל שלו הוא פונק' של הסף. כלומר, בתוך הפונק' נמצא הסף.

$$E = \frac{1}{2}\sum_k (t_k - y_k)^2$$ → k הינו מס' הרמות

$$\frac{\partial E}{\partial w_{jk}} = \Big[\sum_k (t_k - f(y_{in k}))\Big]\cdot \underbrace{\frac{\partial}{\partial w_{jk}}(t_k - f(y_{in k}))}_{\text{נגזרת פנימית}}$$

$K \neq k$ נגזרת שווה ל-0 עבור



$$= f'(y_{in k}) \frac{\partial}{\partial w_{jk}} y_{in k}$$ → הנגזרת שמעניינת אותנו

$$= f'(y_{in k})\, z_j$$

$$\boxed{E = \frac{1}{2}\sum_k (t_k - f(y_{in k}))^2}$$
k הינו מס' רץ וכאשר אנו מסתכלים על k ספציפי אזי עבור שאר ה-k הנגזרת תהא שווה ל-0.

ולכן:

$$\frac{\partial E}{\partial w_{jk}} = [t_k - f(y_{in k})]\cdot -f'(y_{in k}) \underbrace{\frac{\partial}{\partial w_{jk}} y_{in k}}_{\text{נגזרת פנימית של } f' \text{ (באמצע)}}$$

$$= \underbrace{[t_k - f(y_{in k})]\cdot -f'(y_{in k})}_{= \delta_k \text{ נגדיר כ-}} \cdot z_j$$

$$= \delta_k z_j$$

* בעצם האלגוריתם הנ"ל דומה ל-Adeline בשלב 1 בלבד.

$$\frac{\partial E}{\partial V_{IJ}} = \left(\frac{1}{2}\left(\sum_k' (t_k - y_k)\right)^2\right)'$$

$$= \sum_k (t_k - y_k) \frac{\partial}{\partial V_{IJ}} (t_k - f(y_{in\,k}))$$

$$= \sum_k (t_k - y_k) \cdot -f'(y_{in\,k}) \cdot \frac{\partial}{\partial V_{IJ}} y_{in\,k}$$

נדרש רק z_j ספציפי כי השאר נותנים 0

$$= \sum_k (t_k - y_k) \cdot f'(y_{in\,k}) \cdot \frac{\partial}{\partial V_{IJ}} \sum_j z_j w_{jk}$$

לא תלוי ב- V_{IJ}

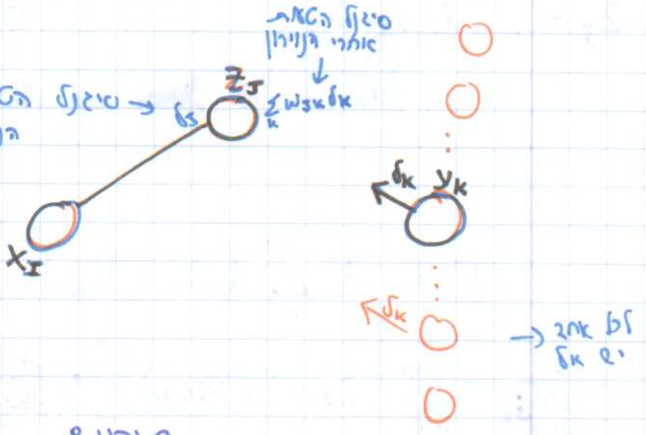
$$= \sum_k (t_k - y_k) \cdot f'(y_{in\,k}) \cdot w_{Jk} \frac{\partial}{\partial V_{IJ}} z_J$$

$$= \sum_k \underbrace{(t_k - y_k) f'(y_{in\,k})}_{\delta_k} \cdot w_{Jk} \frac{\partial}{\partial V_{IJ}} f(z_{inJ}) \qquad f(z_{inJ}) = \sum V_{iJ} x_i$$

$$= \underbrace{\sum_k w_{Jk} \delta_k f'(z_{inJ})}_{\delta_J} X_I$$

$$= \delta_J X_I$$

סיגנל השגיאה לפני הנוירון
הנגזרת של השגיאה - $\delta_J X_I$



* סיגנל טעות יש לנו לפני הנוירון ואחרי הנוירון.

תיקון:

$$w_{Jk}^{new} = w_{Jk}^{old} + \alpha Z_J \delta_k$$

$$V_{IJ}^{new} = V_{IJ}^{old} + \alpha X_I \delta_J$$

בעצם אחרי שמתקנים את W_{JK} מתקנים את V_{IJ}.

האלגוריתם:

רצים בהתחלה קדימה (לאחר שנותנים משקלים התחלתיים) ומקבלים Output.
מוצאים את נגזרת השגיאה ומחשבים שגיאות אחורה ואז מעדכנים את המשקלים וחוזר חלילה.

(*) ב- batch סוכמים את כל ה- δ_k

ב-One at a time צריך לשים לב שמשתמשים בעת חישוב של משקל V_{IJ} במשקל ישן של W_{JK}.

<u>האלגוריתם</u>

- Apply $\vec{x}$ to input unit $(x = x_1 \dots x_n)$ $(x_0 \doteq 1)$
- Calculate for 1^{st} hidden level

$$net_j = \sum_{i=1}^{n} w_{ji} x_i$$

- Calculate output for 1^{st} hidden level

$$O_j = f_j(net_j)$$

- Calculate for 2^{nd} hidden leve

$$net_{j'}^{(2)} = \sum_{j=1}^{1^{st}\ level} w_{j'j} O_j$$

- Calculate output for 2^{nd} level

$$O_{j'} = f_{j'}(net_{j'}^{(2)})$$

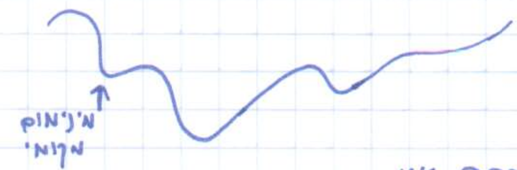
לכל הרמות ⋮

זהו בעצם של move forword

עבור כל k נחשב את
עבור כל j נחשב δ

$$\boxed{*}\ \delta_j \text{ is } \begin{cases} (d_j - O_j) f'(y_j) = \text{for Output} = (d_j - O_j) O_j (1 - O_j) \\ (\sum \delta_k w_{jk}) f'(net_j) \end{cases}$$

בשלב זה נחשב Back E' ~~Forward~~ one at a time



אנו רוצים להימנע ממינימום מקומי

ולכן משתמשים ב-Bipolar Representation

אם נימצא בבור של הפונק' אזי השיפור ייקח הרבה זמן

אם נבחר את הדוגמא רנדומלית כל פעם נימנע מלהיות ממינימום מקומי.

⇐ ל-one at a time יש יתרון על ה-batch כי יש בו רעש.

השאלה - כמה דוגמאות צריך?

- מתי צריך לעצור?

ישנם הרבה פרמטרים חופשיים ברשת (משקלים) ולכן נדרש יותר דוגמאות

אנו מחפשים את ה- # of needed examples

אם נרצה שהטעות תהא e (למשל 10%) אזי (Hasler) טוענים שצריך את

ה- TS $\frac{e}{2}$ גאמ..

→ לדוגמא: יש 80 משקלים ורוצים 10% טעות (אזי נצטרך את הרשת 95 נכונים)

ונצטרך 800 דוגמאות

$$\frac{\#\text{of needed examples}}{\#\text{ of free parameters}} = e$$

$$\#\text{of needed examples} = e * \#\text{ of free parameters}$$

הטעות של Hasler מה שצריך כאמור ויספיקו לרוב.

$$\frac{\#\text{of examples}}{\#\text{of weights}} = \frac{e}{2}$$

$400 = 80 * 5$

BACK-PROPAGATION (עם כל סיגנלי הטעות)
(One-at-a-Time)

- Apply $\vec{X}$ to input units $x=(x_1, \dots x_n)$ $(x_n \equiv 1\ ?)$ → מחשבים Net input קומבינציה לינארית עם המשקלים
- Calculate for 1st hidden level
$$net_j = \sum_{i=1}^{n} w_{ij}\, x_i$$
מחשבים output של הקלט.
- Calculate Output for 1st hidden level
$$O_j = f_j(net_j)$$
(note: f can depend on j!)

- Calculate for 2nd hidden level
$$net^{(2)}_{j'} = \sum_{j=1}^{1^{st}\,level} w_{jj'}\, O_j$$
- Calculate Output for 2nd level
$$O_{j'} = f_{j'}(net^{(2)}_{j'})$$
- etc for more levels

אם יש עוד שכבות נסתרות → מחשבים input ו-output לפי שכבה קודמת

- Calculate Output $O_k = f_k(net_k)$ → כאשר מגיעים לשכבה אחרונה זהו בעצם הפלט של הרשת
- Calculate Error for Output
$$\delta_k = -\underbrace{(t_k - O_k)}_{\text{טעות בין מה שרוצים למה שמקבלים}}\ \underbrace{f_k'(net_k)}_{\text{הנגזרת של הנוירון}}$$
→ חישוב הטעות

כי יש הרבה outputs

[הסקת את הטעות לפני הנוירון]

- Calculate Error for $\underline{\text{Last}}$ Hidden Level
$$\delta_{j'} = f_{j'}'(net_{j'}) \sum_{k}^{Output} \delta_k\, w_{kj'}$$
- Calculate Error for previous Hidden Level
$$\delta_j = f_j'(net_j) \sum_{j'} \delta_{j'}\, w_{jj'}$$

מבצעים קומבינציה לינארית של כל הסיגנלי טעות המשקלים של כל נוירוני השכבה הבאה

→ לוקחים את השגיאה עבור השכבות הנסתרות

- Change Weights:
$$w_{kj}(new) := w_{kj}(old) + \eta\, \delta_k\, O_j$$
$$w_{jj'}(new) = w_{jj'}(old) + \eta\, \delta_{j'}\, O_j$$

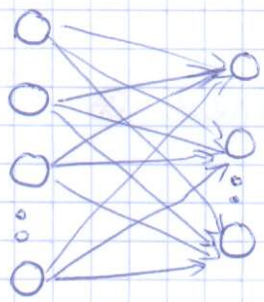
משתנה קבוע

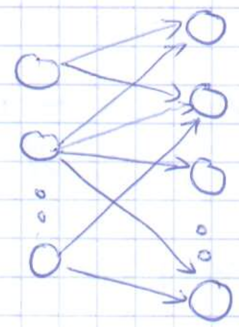
→ חישוב המשקלים החדשים (גם לשכבות נסתרות)

- Use $E = \sum^{P} \sum (t_k - o_k)^2$ to decide ...

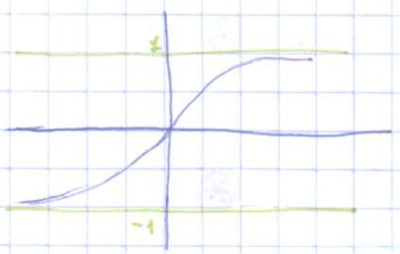
13/6/08

נוירו - חישוביים





* מס' נוירוני הקלט והפלט זהים ~~[illegible]~~
* מס' הנוירונים ב-Hidden Layers יכולים להיות קטנים/שווים/גדולים.



* אנו משתמשים בפונ' אשר מאפשרת רציפות לכן אנו לא חייבים להשתמש בקלט בולאני והפלט יהיה מס' בין 1 ל- -1.

$$f: \{x \mid -1<x<1\}^m \longrightarrow \{y \mid -1<y<1\}^n$$

בעצם אם נשאל איזה פונ' אפשריות אז: all Boolean functions
Continues functions

משפט של Cybenko:

ניתן לבצע קירוב יוניפורמי של כל פונ' רציפה (עד כדי קירוב מסויים) נותנים ε ולכל נתונים הנ"ל תהיה יוניפורמית עד ε.

(*) כמעט כל נוירון

$$f(x) = Sigmoid(x) = \frac{1}{1+e^{-x}}$$
$$f'(x) = f(x)(1-f(x))$$
$$\underset{\text{בעצם}}{\Rightarrow} f(x) = f(net_k)$$

איך ניתן להשתמש ב-B.P.?

* ארכיטקטורה — איך להגדיר את הקלט והפלט?
  - כמה רמות של שכבות נסתרות (ברש)?
  - בכל רמה נסתרת כמה נוירונים יהיו?
  - איזה פונ' יתבצע שימוש
* קלט לימודי — אם יש יותר מידי משתנים אולי יותר קל?
  בד"כ נרצה שהרשת יהיה טוב גם על נתונים חדשים ולכן הקלט הלימודי צריך "לכסות" את כל העולם

* פרמטר לימוד — כמה קטן/גדול אפשר לשנות בזמן לימוד
מכיוון שאנו רוצים ללמוד בזמן סביר אזי נרצה שיהיה adaptivity.

* משקלים התחלתיים — הכי טבעי לאפס אך זה נותן נוירונים סימטריים
ואם שני נוירונים בהתחלה סימטרים הוא ישמור על כך
פיתרון נפוץ - מספרים רנדומליים השאלה - גדולים/קטנים?
תלוי בפונ' סיגמויד - קטן אבל לא נרצה להיות בקצוות
שכן זמן הלימוד יהיה ארוך. (נרצה במרכז).

* מומנטום — אם נשנה משקל מסוים באותו כיוון ⇐ אני בכיוון הנכון ונרצה
(מס' צעדים גדול) ⇔ לשנות בגדול ולכן נסתכל על ההפרש בין המשקלים, אם זה באותו כיוון
נגדיל את השינוי (הוספה של α).

* מינימום מקומי - האלגוריתם הוא ירידה הגרדיאנט ולכן ישנה
מלכודת (מינימום מקומי) ← צריך לטפל: הוספת רעש. ← הסתברות יציאה ממין' מקומי.

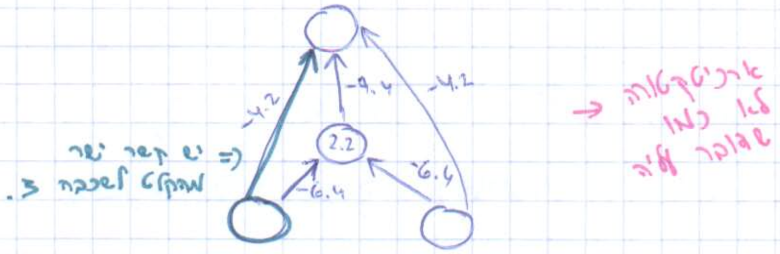
[* ב-Adaline לא היתה בעיה כזו כי הפונ' היא פרבולה = מינימום אחד

* דוגמאות לרשת נוירונים עם שכבה נסתרת לפונ' XOR:

XOR

$\eta = 5$

558 sweeps



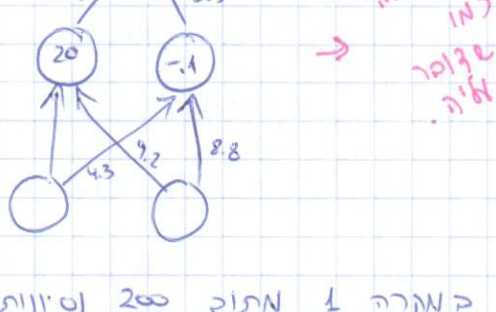
$f(net_k - \theta)$

כאשר משתמשים בסיגמויד הקלט הוא $\sum_{i=1}^{n} w_i x_i$
ואנו רוצים למצוא סף אז נוסיף: במקום $\frac{1}{1+e^{-\sum_{i=1}^{n} x_i w_i}}$ ← $\frac{1}{1+e^{-\sum_{i=1}^{n} x_i w_i + \theta}}$

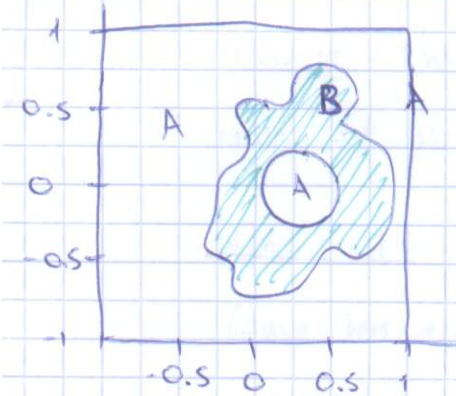
Local Min

$\eta = .25$

10,587 sweeps



במקרה 1 מתוך 200 נסיונות נתקעים במינימום מקומי.



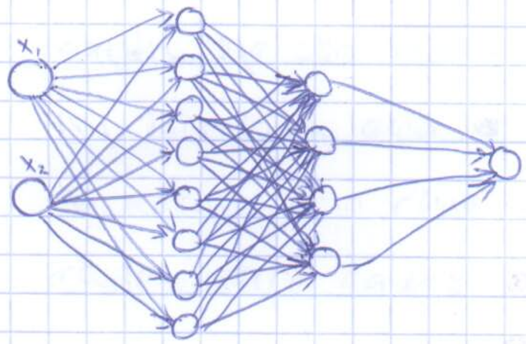
(*) אלמנט הרציפות:

ישנו קלט ממימד 2 (מ- 1- ל- 1)

ולכל קלט שניתן נרצה לדעת האם הוא

A או B.

מסובך מכיוון שיש A בתוך B ו- A מחוץ.



הקלט הוא שני נוירונים

שיקבלו מס' עם מס' ממשיים

בין 1 ל- -1.

2 רמות נסתרות:

רמה נסתרת ראשונה - 8 נוירונים

רמה נסתרת שניה - 4 נוירונים

פלט - נוירון אחד

* כל נוירון ברמה אחת קשור לכל נוירון ברמה הבאה.

(*) היו ניסיונות עם רמה אחת נסתרת שהצליחו.

$f(x)$ = Sigmoid

והתבצע קירוב יוניפורמי.

learning rate = 0.1

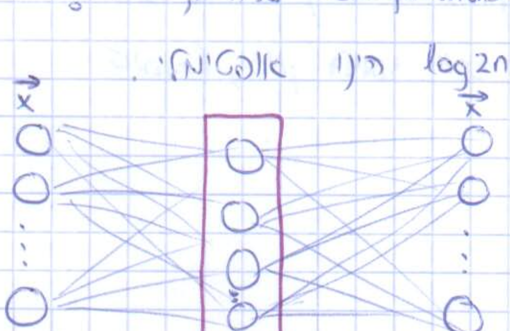
data = 500 נתונים
בפלי' Balance בשני הקבוצות
(250 מ-A ו- 250 מ-B)

* בשכבה הנסתרת השניה לא התקבלו קוים ישרים כמו בשכבה הנסתרת הראשונה אלא מתחילים להוות defining Areas די מסובכות.

* בשכבה האחרונה (פלט) התקבל כיהו'.

(*) דוגמא: נתונים אותיות ורוצים לעשות Compression שלהם, כיצד?

אנו יודעים שאם יש n דברים אזי $\log_2 n$ הינו אופטימלי.



63 נוירונים שמייצגים את הוקטור של כל אות.

רמה נסתרת 1 עם m

קטן מ-63

פלט
רמת (נסתרת) 2 עם 63 נוירונים (שהינה זהה לרמת הקלט).

בעצם הרמה הזו היא קומפרסיה של האינפורמציה

(*) לומדים את פונ' הזהות

(*) כל נוירון קשור לכל נוירון בשכבה הקרובה לו.

(נעשה עם אותיות A-Z בגודל 7x9 המיוצגות ייצוג בינארי)

(*) מכיוון שנעשה שימוש ב-0 ו-1 אזי ישנו בעיה שכן Sigmoid בין 1 ו- -1 ולכן לקחו tolerance של 0.1 ו- 0.2.

<u>המסקנה</u> - כשמוסיפים יותר נוירונים זמן הלימוד יורד (מתחת ל-10 זמן הלימוד גבוה)

וסביר. lot של 0.2 זמן הלימוד קטן, יותר מאשר עבור lot של 0.1.

<u>NetTalk</u> -

(שימוש - לדוג: אנשים בעלי מוגבלות ראיה) text to speach program

phoneme - הברה (לדוג: ח בעברית שאין באנגלית, th שיש באנגלית אך לא בעברית)

באנגלית ישנה בעיה (בעברית אם יש ניקוד אזי אין בעיה) מכיוון שיש דו-משמעות,

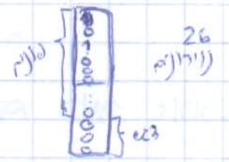
ניתן לבצע שגיאות בביטוי.

הם השתמשו בכוח הלימוד - רשת נוירונים עם רמה נסתרת אחת וביצעו

הלמידה בצורה הבאה: • לאמן מילים בודדות

• לאמן Continues speach

קידוד ראשון (אותיות)



קידוד שני (פונים)

בקידוד 1 את ההופעה כמה נוירונים
שימוש ב- 7 אותיות
26 אותיות באנגלית ועוד . , _
(ולכן 29 נוירונים)

בקידוד 2: 3 לפנים
והשאר ל-phoneme

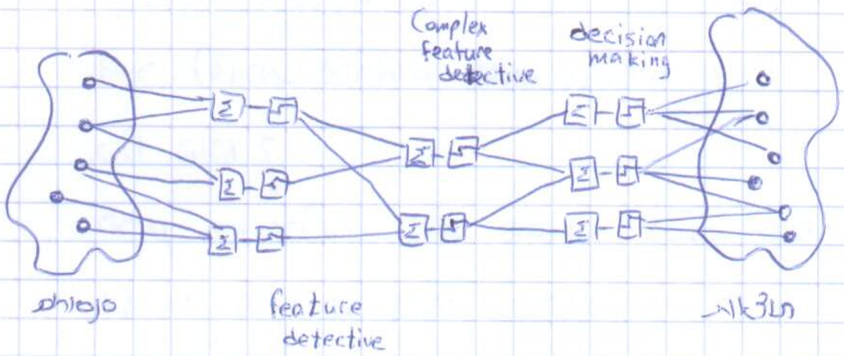
ההצפנה של הקלט הוא מקומי

השתמשו ב-Sigmoid ו- Gaussing Representation

* הם בדקו כמה רגיש להפרעות (רעש),
הם שמו שינויים על המשקלים בצורה רנדומלית
וראו שזה עדיין בטווח (עדיין מניב תוצאות
טובות), זמן הלימוד יותר מהר.

20/6/08

נוירו-חישוביים

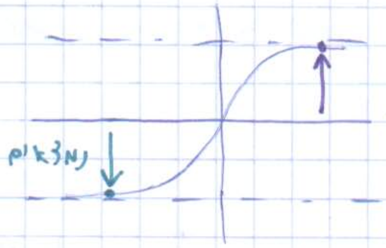


(*) Early Notion of an Intelligent System Built from trainable Perceptron → ארכיטקטורה בסיס

(*) ירידה בגרדיאנט: $w_{ij}^{new} = w_{ij}^{old} - \alpha \frac{\partial Error}{\partial w_{ij}}$

עם המומנטום: (מוסיפים עוד צעד) $+ \beta(w_{ij}^{old} - w_{ij}^{old-1})$ ← מסתכל על השינוי שהיה ומוסיף באותו כיוון

משל:



יקח המון איטרציות.

(הערה: לא בדיוק נכון כי לא מבצעים עבור אחד)

אננו מסתכלים על הפונקציה אלא על פונ' השגיאה (שיכולה להיות ...)

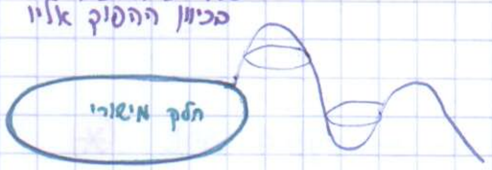
ובה אנו יכולים "להיתקע" במינימום מקומי.

לכן: $\vec{w}^{new} = \vec{w}^{old} - \alpha\, grad(Error(\vec{w})) + \beta(\nabla \vec{w})$

(הצעדים גדולים יותר)

תקי כי הולכים בכיוון ההפוך אליו

רוצים להמשיך באותו כיוון



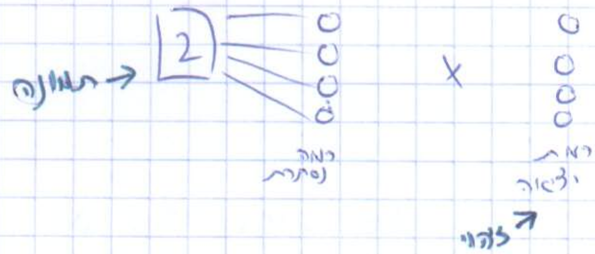
מה קורה אם נהיה בחלק המישורי?

זה אומר שלא משנה מה עושים - אנו רוצים לצאת ממנו. - אם הצעדים יותר גדולים נצא ממנו יותר מהר.

אם בתוך החלק ישנו מינימום מקומי נראה שהסיכוי שניפול לתוכו יהיה נמוך.

(*) כאשר בוחרים מס' נוירונים בשכבה נסתרת ניתן לבחור כמות קטנה ולבדוק את הביצועים ואם לא מספיק טובים להוסיף עוד נוירונים עד שמגיעים לרמת ביצוע מספקת.

(*)



שאלה: כמה טוב יהיה התוצאות? ⇐ 10%

כמה זמן יקח?

* שימוש ב- B.P.

ניסו לזהות Zip Code שנכתב ידני, הם לקחו 12 פיקסלים כאשר כל אחד

64 נוירונים בעצם: 8×8 8×8 ... 8×8 (כל נוירון רואה 25 פיקסלים 5×5)

Weight Sharing - הרעיון

המשקלים המגיעים לנוירון אחד בפילטר מסוים לנוירון אחר באותו פילטר יהיו אותו דבר (למרות שהקלטים ~~[illegible]~~ מאיזורים שונים)

כיצד בוצע?

בזמן האיטרציות בוצע update לכולם ואם אינו זהה מבצעים סכום ~~[illegible]~~ של השינויים.

כלומר, כיוון של פחות משקלים.

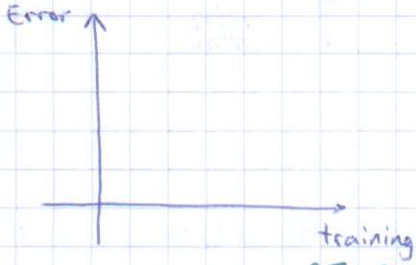
הארכיטקטורה בעצם דורשת שכל הנוירונים האלו (ה-features של אותו) הם מותקים, מבחינת משקלים

ולכן מס' המשקלים קטן יותר

בעצם הארכיטקטורה של התוכנית דומה לארכיטקטורה בסיס (עמוד קודם)

תוצאות: 163,000 דוגמאות של ספרות (חולק ל-training ו- test)

23 epochs ~~[illegible]~~ ← לקח 3 ימים (כיום לוקח מס' דקות)



training: 2.5×10^{-3}

test set: 1.8×10^{-2}

ב-training היה 0.14% misclassification

ב- test 5% misclassification ⇐ הצלחה של 95%.

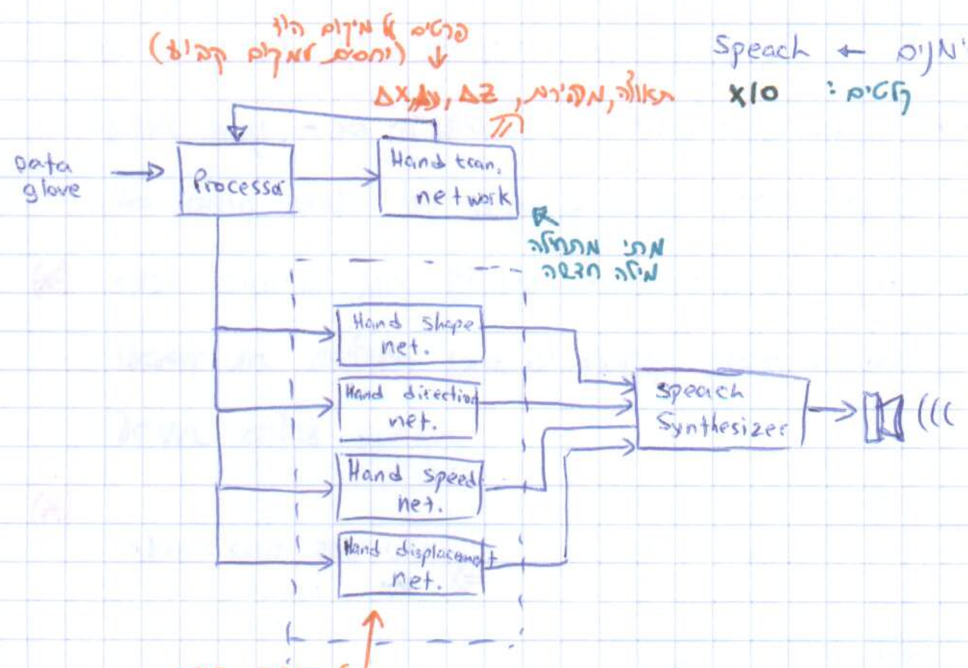
האלגוריתם צריך להחזיר תשובה - זה הספרה או לא בטוח ולתת 2 אפשרויות

↑ שיפור

(*) כדי לקבל 99% הצלחה צריך לעשות rejection של 12%. (כלומר 87% בטוח 12% לא ו-1% לא יודע)

(*) שפת סימנים ← Speach

קלטים: x10



(*) 50 קלטים סה"כ.

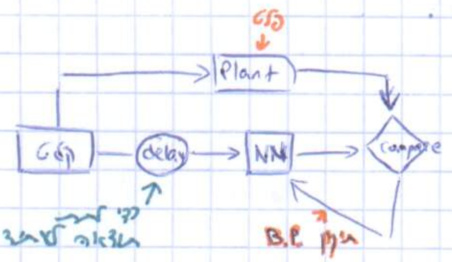
בעיה ב- Hand tran. net. משפיעה על השאר

(*) כל רשת נלמדה כתפקיד בנפרד.

תוצאות: 99% אולם 5% כאשר אין output (כאשר לא יודעים מתי מתחילה ומסתיימת)

(*) מצגת: NNGeneralOverView

ניתן להשתמש ברשת כדי לנבא.



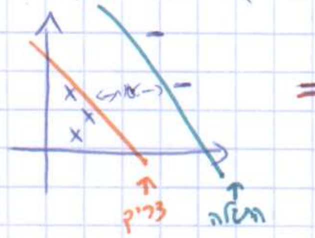
לדוגמא: בורסה - קלט הינו הנתונים של היום
מה תהיה ה-k מחר
ואז להשוות האם הניבוי היה
נכון.

(*) מצגת: Bottleneck N.N.

מתחילים בקלט מסוים ומנסים ללמוד את פונ' הזהות ⇐ שמים צוואר בקבוק (שכבה נסתרת עם מס' נוירונים קטן יותר מקלט)

הקלט הינו $\mathbb{R}^n$ השכבה הנסתרת $\mathbb{R}^k$ (כאשר $k<n$) והפלט $\mathbb{R}^n$.

לפי עקרון שמבק הונים לא ניתן ללמוד את פונ' הזהות אבל כאן איננו לומדים מפקטים. כי מנת ללמוד אנו צריכים positive data ו-negative data הבעיה היא שה-negative data צריך להיות מיוצג (אחרת יכול להיות שהשגיאה תהיה גדולה מידי)

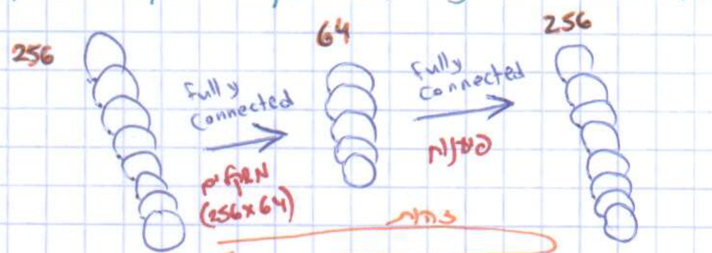


⇐ הסיבה היא שה-negative data לא היה מיוצג.

דוגמא: הקלט יהיה תמונות, נניח.

הקלט החדש - אם דומה לזהות הוא דוגמא חיובית אחרת ~~לא~~ דוגמא שלילית.

← ה-BottleNeck יהיה כיווץ של התמונה (למשל לקחת חלק מהתמונה)



* התמונה של Lena
ניקח חלקים בגודל 16x16
* ההצפנה של החלקים הוא ההצגה של התמונה
* לצבוע כל תת-חלק המשקלים זהים.

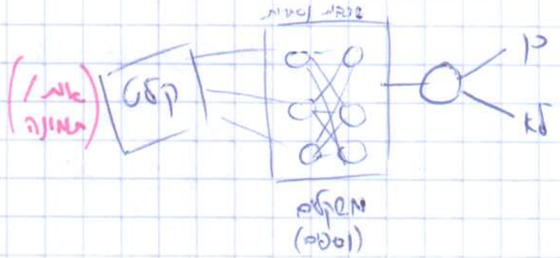
* אותו רשת מבצעת Compression ופענוח ← יש כיווץ פי 4.

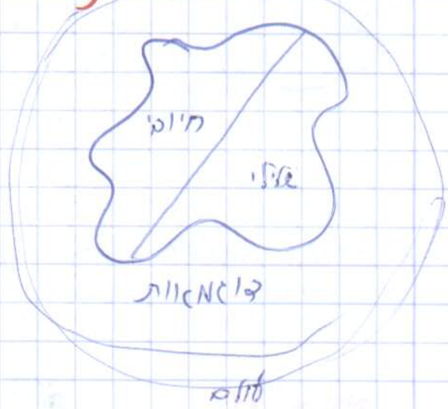
הערה: הפעילו עם תמונה של Amal (ציירה). כלומר, Lena ו-Amal זהות (מבחינה ששניהם תמונות של אנשים) אם נשים תמונה של RaceCar אולי לא נזהה ⇐ פילטר המזהה סוג של תמונות (נניח למצוא עור)

27/6/08

נוירו - חישוביים

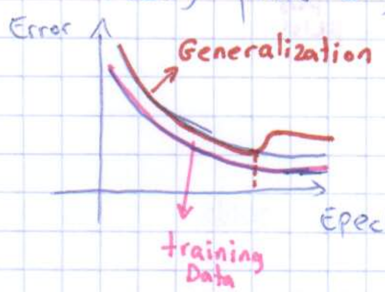
2-class labeled training data





המטרה לענות על דברים אחרים- נכון או לא נכון (לאיין דוגמאות חדשות הסכנה: over trained.

כלומר, השאלה- מתי נפסיק לבצע לימוד? כיוון שלימוד יתר יכול לגרום שנצליח 100% עבור ה- data שלמדנו אבל עבור ה-data שנדרש למבחן נכשל.



— דוגמאות של דברים דומים

<u>n-fold</u>

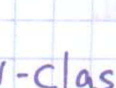
- לערכן את הרשת
- לבדוק כמה הרשת טובה

נבצע ממוצע ⇐

חילקנו את ה- data ל- 10 (10-fold) נבדוק 90%.

* ל- 10% נבחן.
* לאחר מכן נבחן 10% שונים וכך הלאה...

1-class training



→ לומדים את פונ' הזהות (כאו ~~enna~~) אם קרוב לפונ' זהות חיובי אחרת שלילי.

N-Class - FMRI
specification

כיצד נדע מה הפעולה שאדם עושה ע"י הסתכלות ב-MRI.

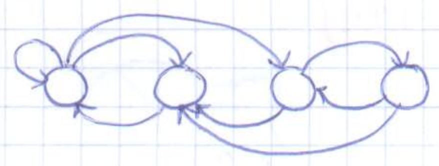
לפי תמונות - על מה אדם מסתכל מה- MRI שלו.

תוצאות: 90% ניתן להבחין בין Blank ל- faces (2-class) (וכו' לכל ה-classes)

80% עבור 1-class (שימוש ב- wrapper לצמצום הרעש)

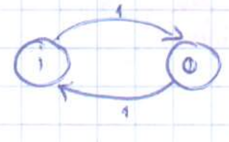
תרגיל 2 - הגשה 11/7.

מה קורה כשיש לולאות?



למשל במוח ישנם דברים הנראים -

נסתכל על 2 נוירונים:



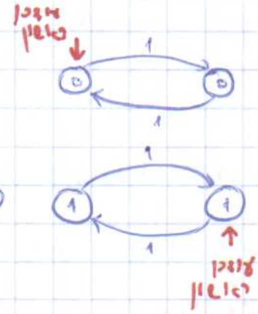
מה יקרה? סינכרוני - יתבצע update לשניהם באותו זמן
א-סינכרוני - יתבצע update לאחד ולאחר מכן לשני

הסף = 0

סינכרוני:

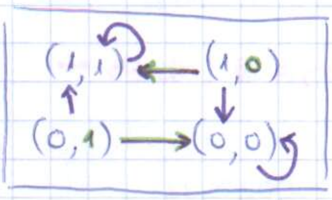
→ לא תהיה התכנסות

א-סינכרוני:



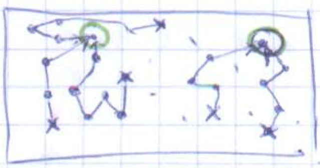
⇐ בשלב הבא לא ישתנה → הגענו למצב יציב

⇐ בשלב הבא לא ישתנה



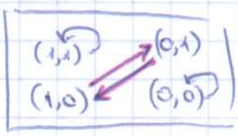
→ עבור 2 נוירונים ישנם 4 מצבים.

כלומר, ישנם 2 מצבים יציבים ו-2 לא יציבים.



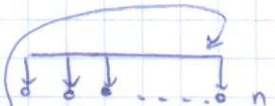
טענה - במקרה של הרבה נוירונים אזי יש תהליך שמביא אותנו למצ' יציבות.

במקרה זה הנוירונים היציבים הינם זכרונות והמצב ההתחלתי יגיע לזיכרון בצורה "אסוציאטיבית".



צריך להיזהר ממצב שלא מגיעים ממנו למצב יציב (-לולאה בין 2 מצבים)

מטריצת משקלים



רשת של נוירונים כאשר כל נוירון קשור לכל נוירון

קשרים בין נוירונים לנוירונים → $W^n(\ \ ) \leftarrow n \times n$

$W^n(\ ^m) \leftarrow n \times m$

→ גרף דו-צדדי של נוירונים

זיכרון יהיה שמור אם:

$W\vec{x}^1 = \vec{x}^1$

$W\vec{x}^2 = \vec{x}^2$

$W\vec{y} = W(\vec{x}^1 + \vec{e}) = W\vec{x}^1 + W\vec{e}$

$\vec{e}$ שגיאה

$(\vec{e} \to 0 \Rightarrow W\vec{e} \to 0)$

* יגאל אומר -
אם נתחיל ממנו נישאר בו

(*) כיצד נמצא את המשקלים?

נתונים n וקטורים אורתוגונליים: $\vec{S}^1, \dots, \vec{S}^n$

$$\vec{S}^i \cdot \vec{S}^j = 0 \quad i \neq j \quad \Longrightarrow \quad W = \sum_i \vec{S}^{i\,T} \vec{S}^i$$

outer product: $n(\ ^1) \times (\ ^n) = n(\ ^n)$

בעצם:

$$W = \begin{pmatrix} & & \\ & w_{ij} = s_i t_j & \\ & & \end{pmatrix}$$

$\vec{s} \quad \vec{t}$

$\vec{S}^{①} \quad \vec{S}^{②}$

מה יקרה אם ניקח את W ונכפול עם $\vec{S}^{①}$?

נניח והיו לנו 2 וקטורים $\vec{S}^{①}$ ו-$\vec{S}^{②}$ אזי:

$$\vec{S}^{①} W = \vec{S}^{①} \left(\vec{S}^{②T} \vec{S}^{①} \right)$$

if $\vec{S}^{①} \perp \vec{S}^{②} \Rightarrow 0$

אורתוגונלי

היינו חושבים ב-0

לא בדיוק נכון אולי

$$\vec{S}^{①} \left(\vec{S}^{①T} \vec{S}^{①} + \underbrace{\vec{S}^{②T} \vec{S}^{②}}_{0} + \underbrace{\vec{S}^{①T} \vec{S}^{①}}_{0} \right)$$

$$\|\vec{S}^{①}\|^2 \vec{S}^{①}$$

$$w_{ij} = \sum_{k=1}^{m} \sum_{l=1}^{n} S_i^{(k)} \cdot S_j^{(l)}$$

עבור $m=2$:

$$W = \left(S^{①T} \cdot S^{①} + S^{②T} \cdot S^{②} + \underbrace{S^{①T} S^{②}}_{0} + \underbrace{S^{②T} S^{①}}_{0} \right)$$

$$\vec{S}^{①} \cdot w = \|S^{②}\|^2 \cdot \vec{S}^{①} + \|S^{①}\|^2 \cdot \vec{S}^{①}$$

$$= \vec{S}^{①} \left(\underbrace{\|S^{①}\|^2}_{1} + \underbrace{\|S^{②}\|^2}_{1} \right)$$

(מנורמלים)

$$= \vec{S}^{①}$$

Grossberg - Hopfield Networks (Auto-Associative)

BAM network (Binary Associative Memory)

n Mc-Pitts (bipolar)

n נוירונים עם קשירה מלאה (כולם קשורים לכולם)

$\vec{S}^i = \{1, -1\}^n$

State

משקלים קבועים $W = {}_n(w_{ij})^n$

אקטיבציות ← איך הרשת עובר בין ה- States

לאיפה הרשת תגיע ממצב התחלתי מסוים?

$$S_i = \text{sign}\left(\sum_j w_{ij} S_j - \cancel{\theta} \right)$$

הנוירון ה-i

אם θ שווה 0

הנחה: $w_{ij} = w_{ji}$

נבצע עדכון א-סינכרוני

ניקח $\vec{t}$ ונרצה שיהיה יציב

$$\text{sign}\Big(\sum_j w_{ij} t_j\Big) = t_i$$

או ±

רוצים ל"שמור" רק תבנית אחת $\vec{t}$

והשאלה איזה w_{ij} ניקח?

$$w_{ij} \propto t_i t_j$$

$$\text{sign}\Big(\sum_j (\underbrace{t_i \overbrace{t_j}^{1}}_{w_{ij}} t_j)\Big) = \text{sign}\Big(\sum_j t_i\Big) = \text{sign}(n t_i) = t_i$$

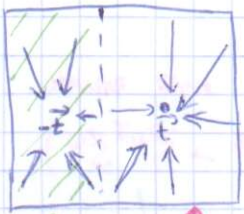
t_i לא תלוי ב-j

(נזכור כי: $\forall i \; t_i^2 = 1$)

לא איבר במקום i

$$net_i = \sum_j w_{ij} t_j = \sum_{j \text{ "נכון"}} t_i t_j^2 - \sum_{j \text{ "לא נכון"}} t_i^{wrong} t_j^2$$

קרוב ל-t ; לא קרוב ל-t



לא נכון, יותר רחוק מ-½ "ההפוך" ל-$\vec{t}$-

קרוב ל-$\vec{t}$ "ההפוך" ל-$\vec{t}$-

⇐ כלומר, אני אקבל תשובה ב"נכון" אם רוב הביטים "נכונים"

⇒ attraction

⊛ בעצם מכיוון שאני "שומרים" רק תבנית אחת אזי היא attractor (מינימום יציב)

⊛ מה אם נרצה לשמור יותר מדפוס אחד? (p תבניות)

$$W_{ij} = \sum_{p=1}^{P} t_i^p t_j^p$$

רצים על תבניות

$$t_i^{p_0} \stackrel{?}{=} \text{sign}\Big(\sum_j w_{ij} t_j^{p_0}\Big) \quad \longleftarrow \text{לתבנית ספציפית}$$

$$t_i^{p_0} = \text{sign}\Big(\sum_j \Big(\sum_{p=1}^{P} t_i^p t_j^p\Big) t_j^{p_0}\Big) = \text{sign}\Big(\sum_j \sum_{p=1}^{P} t_i^p t_j^p t_j^{p_0}\Big) =$$

נפתח

$$= \text{sign}\Big(\sum_j t_i^{p_0} + \underbrace{\sum_{p \neq p_0} t_i^p t_j^p t_j^{p_0}}_{\text{"crosstalk"}}\Big) = \text{sign}(n t_i^{p_0}$$

עבור $p = p_0$

במקרה שהם אורתוגונלים יהיה שווה ל-0 (uncorrelated)

⊛ אם הם מנוגדים (או קרובים לכך) נגיע למס' גדולים מאד n.

השאלה - כמה P? אם אורתוגונלים נרצה הרבה אחרת לא.

צריך לדעת מקס' של P כך שה-crosstalk יהיה קטן. → עוזר לנו

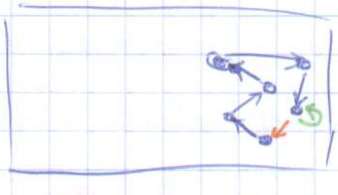
נזכור שה-sign

שאלה-

כמה P אפשר לשמור, האם יש Convergence?

⊛ האם ימשיך עד לאינסוף?

או ⊛ "יתכנס" ל-attractor?



4/7/08

נוירו - חישוביים

רשת נוירונים עם לולאות:

$$W := \underbrace{\vec{x}_1^T\vec{x}}_{} + \vec{x}_2^T\vec{x} + \dots + \vec{x}_n^T\vec{x}$$

מטריצה סימטרית

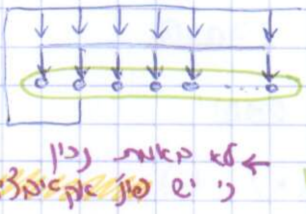
$$\omega_{ij} = \omega_{ji}$$

במקרה שהוקטורים אורתוגונליים:

$$\vec{x}_3 W = \underbrace{\vec{x}_3\vec{x}_1^T}_{0}\vec{x}_1 + \underbrace{\vec{x}_3\vec{x}_2^T}_{0}\vec{x}_2 + \underbrace{\vec{x}_3\vec{x}_3^T}_{n}\vec{x}_3 + \dots$$

← הכפלה פנימית

⇐ התוצאה תהא $\vec{x}_3$, שומר זיכרון



בעצם ω_{ij} הינו משקל בין i ל-j.

$\vec{x}$ $\vec{x}$

$W\vec{x}$ בעצם $\Phi(W\vec{x})$

$W(W\vec{x})$ $\Phi(W(\Phi(W\vec{x})))$

← לא מאומת נכון כי יש פונ' אקטיבציה

אם לא אורתוגונליים:

$w(\vec{y}) = w(\vec{x} + \vec{e}) = W\vec{x} + W\vec{e}$ ⇐ אם e קטן אז נקבל משהו קרוב ל-$\vec{x}$.

ב' אפשרויות: סף "מכסה" את השגיאה → $\Phi(W\vec{x} + W\vec{e}) = \vec{x} + \tilde{e}$

ישנה התכנסות כתוצאה מאיטרציות

אם היינו יכולים להגדיר את הרשת כך שהם מקבלים את ה-$\vec{x}$ שהם נקודות שבר (מכיוון שיש לנו סף ואיטרציות) האם תהיה לנו תמיד התכנסות?

Hopfield טען:

אם ניקח שני וקטורים רנדומליים הסבירות שהם יהיו אורתוגונליים היא נמוכה (שואף ל-0)

$\vec{r}_1 = (1, -1, 1, 1, -1, \dots)$

$\vec{r}_2 = (-1, 1, 1, -1, 1, \dots)$

← הסבירות שהם יסכימו על אחד $\frac{1}{2}$
הסבירות שהם לא יסכימו $\frac{1}{2}$
לכן אם n מספיק גדול הם יהיו קרובים לאורתוגונליות.

$\underset{n\to\infty}{E}(\vec{r}_1 \cdot \vec{r}_2) = \vec{0}$ ← אם n מספיק גדול

$x_1, x_2, \dots, x_k$ ← לא תלוי ב-n

$W := \frac{1}{n}\sum \vec{x}_i^T\vec{x}_i$ [ ] ← אני לא מנסה שיהיה שווה ל-0 אבל מכוון שיש גם סף וגם איטרציות

$$\vec{x}_j W = \frac{1}{n} x_j x_j^T x_j + \underbrace{\sum x_j x_i^T x_i}_{\text{"Crosstalk"}}$$

כאשר מסתכלים על 1:

→ $\vec{v}$ זיכרון וגם $-\vec{v}$ שניהם attractors

כאשר מסתכלים על יותר מ-1 נראה:

→ האם זה באמת כך?

משפט: תנאים → (Completely Connected , bipolar , $M_c \cdot P$ neurons) [asynchronous update]

כל פעם נוירון אחד

* משפט ההתכנסות:
אם נתחיל מהפרמטרים שנתנו
אז נגיע למצב יציב

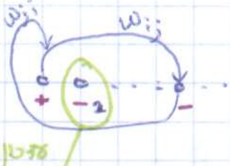
(i) (Symetric) $w_{ji} = w_{ij}$

(ii) $w_{ii} = 0$

ממצב התחלתי כלשהו נגיע למצב יציב.

הסבר:

[דוגמא: $\vec{W} = \sum_{x} \vec{x}^T \vec{x}$]



$X_{new} := \Phi(net_{in}) = \Phi(\sum W_{i2} X_i)$

התוכן של הוקטור יכול להשתנות אולם מספר הפעמים שהוא ישנה הינו סופי ולכן נגיע למצב יציב.

הוכחה:

נמצא פונ' בעלת 3 תכונות תגרום להתכנסות (אם קיימת הנ"ל) → $Energy(\vec{x})$

(i) $E(\vec{x})$ מוגבלת מלמטה

(ii)* כל שינוי ב-E משינוי ב-$\vec{x}$ פירושו מקטין את $\vec{E}$.

(iii) קיים $\delta > 0$ כך שכל שינוי הוא לפחות δ.

ניקח את הפונ': $E(\vec{x}) = -\sum_{i,j} x_i w_{ij} x_j$ (נוירונים של המצב) → אם הסף הוא 0 (אם לא נדרש להוסיף משהו עבורו.)

* אם $w_{ij} = 1$ (חוץ מ- $w_{ii} = 0$)
* אם $x_i = 1$ אז נקבל $E(\vec{x}) = -n^2$ (⇐ יש מינימום)

$$\Phi(x_j) = \begin{cases} -1 & \sum w_{ij} x_i < \theta \\ x_j & \sum w_{ij} x_i = \theta \\ 1 & \sum w_{ij} x_i > \theta \end{cases}$$

הגדרת פונ' האקטיבציה

$E(\vec{x}) = -\sum_{i,j} x_i w_{ij} x_j + \sum_i \theta_i x_i$ — אם הסף לא 0

הוכחת תכונות:

(i) $E(x) \geq -\sum |w_{ij}|$ ← אם סף לא 0 אז מוסיפים $-\sum_i |\theta_i|$

(iii) טריוויאלי יש מס' שינויים אפשריים כל שינוי יקטין בגודל מסוים (פעם אחד) ולכן ניקח את המינימום של כולם.

(ii) $E^{new} - E^{old}$ → רוצים להראות שזה תמיד שלילי

$$= -\sum_{i,j} x_i^{new} w_{ij} x_j^{new} - \left(-\sum_{i,j} x_i^{old} w_{ij} x_j^{old}\right)$$

* asyncrony: עדכון של x_I, כלומר, $x_j^{new} = x_j^{old}$ כאשר $i \neq j$; * $w_{ii} = 0 : \forall i$

$$= -\sum_j x_I^{new} w_{Ij} x_j + \sum_j x_I^{old} w_{Ij} x_j$$

$$= \left(-x_I^{new} + x_I^{old}\right)\left(\sum w_{Ij} x_j - \theta_i\right)$$ net in (I)

$$E = \frac{A}{2}\sum_x{}' \sum_i{}' \sum_{j\neq i}{}' V_i^x V_j^x$$

אותה שורה

כמה לא מסדר אם יש יותר מעיר אחת מודלקת באותה שורה

L^x

$$+ \frac{B}{2}\sum_i{}' \sum_x{}' \sum_{y\neq x} V_i^x V_i^y$$

אותה עמודה

כמה לא מסדר אם יש יותר מעיר אחת מודלקת באותה עמודה

$$+ \frac{D}{2}\sum_x \sum_{y\neq x} \sum_i d_{xy} V_i^x \left(V_{i+1}^y + V_{i-1}^y\right)$$

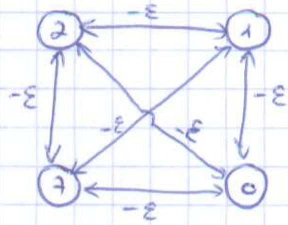
כמה לא מסדר אם המרחק גדול.

$$+ \frac{C}{2}\left(\left(\sum_x\sum_i V_i^x\right) - n\right)^2$$

כמה לא מסדר אם לא "בחרנו" אף עיר.

(*) המשקלים קבועים ואני מחפשים מינימיזציה של הבעיה בעזרת E. ← אופטימיזציה

__בעיית מציאת מקסימום:__

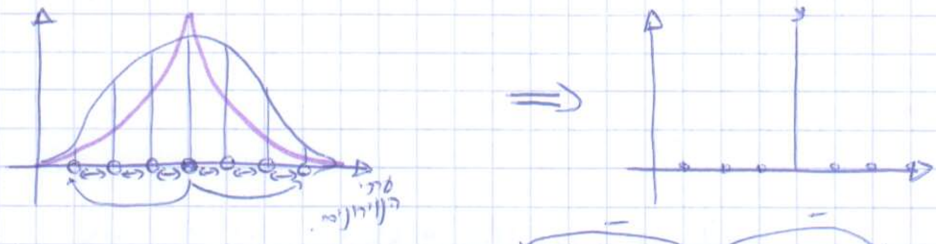


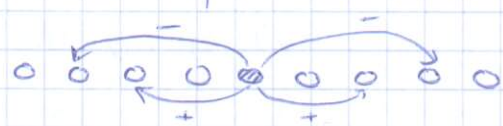
ישנה תחרות בין הנוירונים

ורק אחד "ישאר".

* כאשר נירון מגיע ל-0 הוא כבר לא משפיע

בעצם בכל צעד המקומות מוקטנים (ובסוף ישאר רק נירון אחד הגדול מס' צדדו מ-0)





: Sharpening

"נספק את העסק" כך שקרובים יהיו חיוביים ורחוקים יהיו שליליים.

נרצה להראות שזה שלילי, כלומר ~~ישנם~~ ישנם שני מקרים:

<u>מקרה I:</u> $X_I^{old} = 1$ $\quad X_I^{new} = -1$

$net_{in(I)} < 0$ $\quad -X_I^{new} + X_I^{old} > 0$

$\Leftarrow$ המשוואה שלילית. $\quad \nabla E < 0$

<u>מקרה II:</u> $X_I^{old} = -1$ $\quad X_I^{new} = 1$

$net_{in(I)} > 0$ $\quad -X_I^{new} + X_I^{old} < 0$

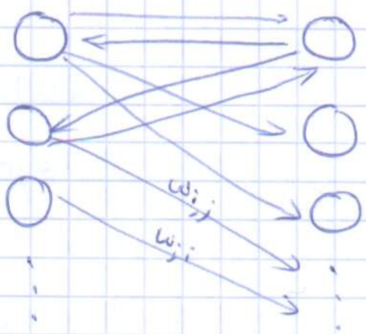
$\Leftarrow$ $\quad \nabla E < 0$

- auto-associative ⊛

כמה דוגמאות אפשר לשמור/לזכור עבור 100 נוירונים?
תשובה- 14 בערך.

BAM

(Binary Associative Memory)



שתי דו-כיווני עם קשר מלא בין השכבות, בלי קשר בתוך כל.

<u>תרגיל בית</u>

נשמור $(\vec{x}, \vec{y})$

$$W = \sum_{<\vec{x},\vec{y}>} \vec{x}\,\vec{y}^{T}$$

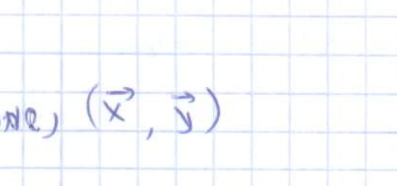
$n\begin{pmatrix} & m & \\ & & \end{pmatrix}$

אם ניקח את הפרמטרים הללו נגיע לנק' יציבה

<u>נדרש</u>- לשנות את E להתאים ל- y, x.

TSP

ישנו איש מכירות האמור לטייל ב-n ערים ורוצים שהזמן יהיה מינימלי.



יש לנו n^2 נוירונים ונרצה שבכל עמודה וכל שורה "ידלק" נוירון אחד

נגדיר משקל שלילי בין כל נוירון ונוירון בעמודה.

נגדיר משקל שלילי בין כל נוירון ונוירון בשורה.

נגדיר מרחק בין נוירונים בשורות ועמודות שונות (כלומר, נגדיר משקל חיובי)

$V_i^x \rightarrow$ מגדור נוירון בשורה x ועמודה i.

11/7/08

נוירו-חישוביים

תרגיל בית 2:

- אפשר לבדוק בין הטעות של דחיסה של תמונות שונות.
- נדרש לבדוק האם האלגוריתם מבצע דחיסה ל-Text

* בעצם בוצע - (1) דחיסה/הצפנה

(2) פילטר לתמונות

הערה - השתמשנו ב-one class, אם היינו משתמשים ב-two class (1 לתמונה ו-2 ל-text מסוים) אזי היינו יכולים לקבל קלט שאינו שייך לאף אחד מה-classes ולא היה יודע מה להחזיר ← הוא יבדוק לאיזה class הוא דומה יותר.

↳ קשה להגדיר trainning set שיכיל את הדברים השליליים

B.A.M

תרגיל בית 3 - מימוש associative memory בעזרת BAM

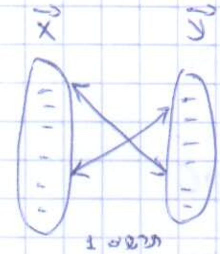
מס' טלפון - 5678521

(א) הגדרה כ-attractors ובקבלת מס' טלפון 5678421 יחזיר attractor

← fault tolerance

(ב) נתונים ומס' טלפון icons בגודל 4x4

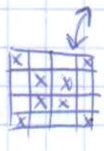
$\vec{y}$ ← 5678521



תרשים 1

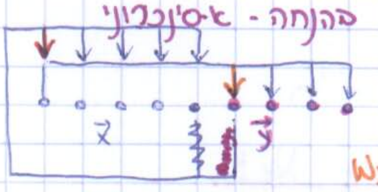
$$W = \sum_{p} \vec{x}^{p} \vec{y}^{pT}$$

דוגמא p



$(w_{ij} = w_{ji})$

הנחה - א-סינכרוני



תרשים 2

בשביל לעשות נדרש למלא את פונ' האנרגיה.

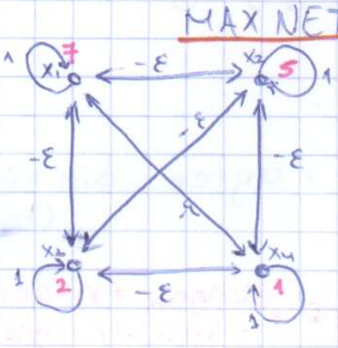
$W_{y_1 x_1} = W_{x_1 y_1}$

$W_{x_1 y_1} = W_{y_1 x_1}$

[ניתן להסתכל על תרשים 1 בתצורה של תרשים 2]

רשתות עם תחרות

MAX NET



$\varepsilon = 1/4$

הנוירון נותן $\sum w_i x_i$ (← פונ' activation)

הנחה - אין 2 נוירונים אותו דבר.

$$X_j^{new} = \sum_{i \neq j} w_i x_i^{old} + x_j^{old}$$

בכל שלב הנוירון יקטן. תנאי עצירה שלם 0 חוץ מ-1 N

שלב 1:

$X_1^{new} = X_1^{old} + \sum_{i \neq j} w_i X_i^{old} = 7 - \varepsilon (5+2+1) = 5$

$X_2^{new} = 5 - \varepsilon (7+2+1) = 2.5$

$\left.\begin{array}{l} X_3^{new} = 2 - \varepsilon (7+5+1) < 0 \\ X_4^{new} = 1 - \varepsilon (7+5+2) < 0 \end{array}\right\} \rightarrow$ כבר לא מעניינים בשלב הבא

שלב 2:

$X_1^{new} = 5 - \varepsilon \cdot 2.5 \approx 4.4$

$X_2^{new} = 2.5 - \varepsilon \cdot 5 = 1.25$

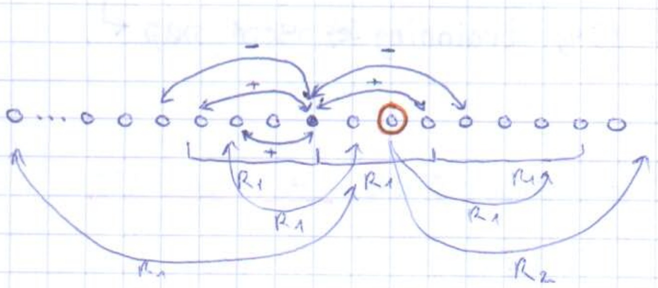
שלב 3:

$X_1^{new} = 4.4 - \varepsilon \cdot 1.25 = 4.1$

$X_2^{new} = 1.25 - \varepsilon \cdot 4.4 \approx 0.1 \rightarrow$ בשלב הבא יתכנס ל-0

$\Leftarrow$ המקסימום הינו X_1.

Mexican Hat



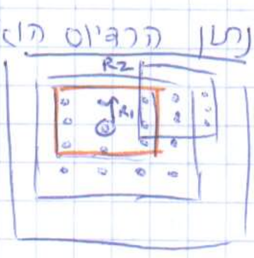
כו תמונה לנוירון 1, אותו דבר קורה בדיוק עם המשקלים האחרים.

$\rightarrow$ בעצם מתרחש תהליך של Sharpening

יוצר פונ' "חידוד".

טופולוגיה חיצונית - נתון הרדיוס הוא לא חלק מהמשקלים/נוירונים

אפשר בדו-מימד:



Hamming Net

בסוף רק אחד יהיה מודלק
מה שאומר שהכי קרוב לאב-טיפוס הנ"ל.

את המשקלים אנחנו נשים באותו ייצוג כמו ה-input

$\vec{w}^{(i)} \vec{v} = agree - disagree$
(hamming distance)

Bipolar

נזכ' כי: $n = agree + disagree \Rightarrow disagree = n - agree$

ולכן: $\vec{w}^{(i)} \vec{v} = agree - (n - agree)$

$$\vec{w}^{(i)}\vec{V} = 2\,\text{agree} - n \qquad \Leftarrow$$

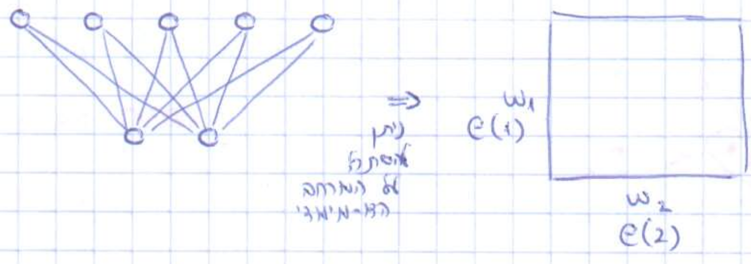
$$\text{agree} = \frac{\vec{w}^{(i)}\vec{V}}{2} + \frac{n}{2}$$

$\underbrace{\vec{e}_1, \vec{e}_2, \dots, \vec{e}_7}_{\text{אבטיפוס}}$

נבחר את המשקלים להיות האבטיפוס חלקי 2.

ונשים Bias : $n/2$

מרחב המשקלים



(המקום הגיאומטרי של המשקלים)

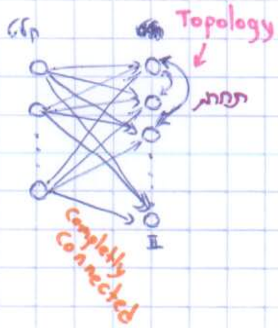
* מתארים את המשקלים לפי המקום הגיאומטרי שלהם. וזה מייצג את הנוירון.

במרחב המימד המקורי של המשקלים

2-class data	+	-
1-class data	+	
0-class data		

0-class - יש רק data ולא ידוע לאיזה class שייך.

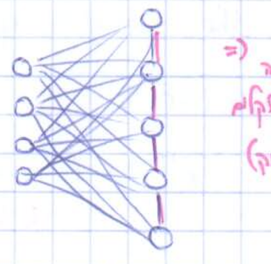
למה זה טוב?



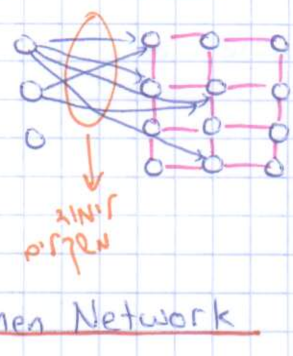
ניקח קלט שבו כולם קשורים לכולם ונוסיף את האלמנט של תחרות.

כלומר, שמים Topology על רמה II

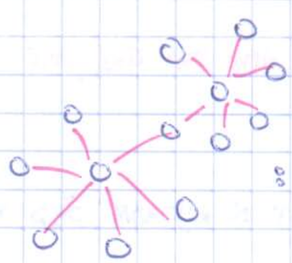
למשל: טופולוגיית קו:



טופולוגיה דו-מימדית:



ניתן גם להסתכל כך:



Self Orginizing Neural Network : Kohonen Network

(1) $\vec{X}$ קלט

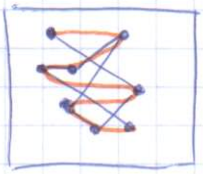
(2) Find closest $\vec{z}$ to $\vec{X}$ in מרחב המשקלים in kohonen level

learning_

(3)

הרעיון - Clustering

למשל של טופולוגיית קו :

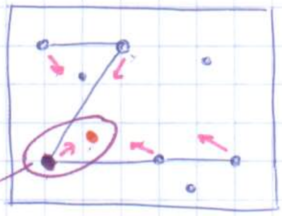


בהתחלה נבחר רנדומליות ולכן לא בהכרח "מסודר". נרצה שבסוף יהיה מאורגן.

אם נסתכל על בעיה שלב (ב 2 צעדים)

• - נקודה חדשה

← קירוב



* נקרב את כולם לנקודה.

נוירון הכי קרוב

בעצם נחפש את הנוירון הכי קרוב

אחרי ההתכנסות של האלגוריתם התמונה תהיה מסודרת.

$$\{ \vec{x} \mid \min \| \vec{u} - \vec{x} \| \}$$

$N(\vec{x})$ - neighborhood

α - learning parameter

Algorithm

(1) בחר נתון $\vec{u}$ (קלט)

(2) find closest $\vec{z}$ to $\vec{u}$ (By Hamming net or min Euclidian distance)

* in the weights space in kohonen level

(3) Learning:

$$\vec{x}_i^{new} = (1-\alpha)\vec{x}^{old} + \alpha \vec{u} \quad ; \quad \vec{x} \in N(\vec{z}) \text{ לכל}$$

← מזיזים בכיוון $\vec{u}$

(4) update N (בהתחלה N גדול) → לאט, לאט יהיו פחות שכנים

(5) update α

* בד"כ מעדכנים N יותר מהר מ-α.

עוצרים כשהשינויים כבר קטנים.

18/7/08

נוירו-חישוביים

(*) ב- Hamming Net האבטיפוס ייוצגו ע"י משקלים לוקטור הקלט כאשר
מס' משקלים תלויים בממד וקטור הקלט.

(*) אם אין אבטיפוס מראש?
נניח שיש 7 כאלה, נבחר משקלים רנדומליים ואז לאט, לאט הקלטים
יתקבצו ל-7 קבוצות.

בעיית ה- Uniform / non-Uniform

צריך שהתפלגות ה-clusters יהיה שונה מהתפלגות ה- data
← Equi Probable Map

לכן, במקרה שה- data לא מתפלג אחיד אזי יהיו cluster
שיבחרו הרבה פעמים ← נדרש לתת להם קנס על מנת שאחרים ייבחרו.

$$P_j^{new} = P_j^{old} + \beta\left(y_i - P_j^{old}\right)$$

$0 < \beta < 1$

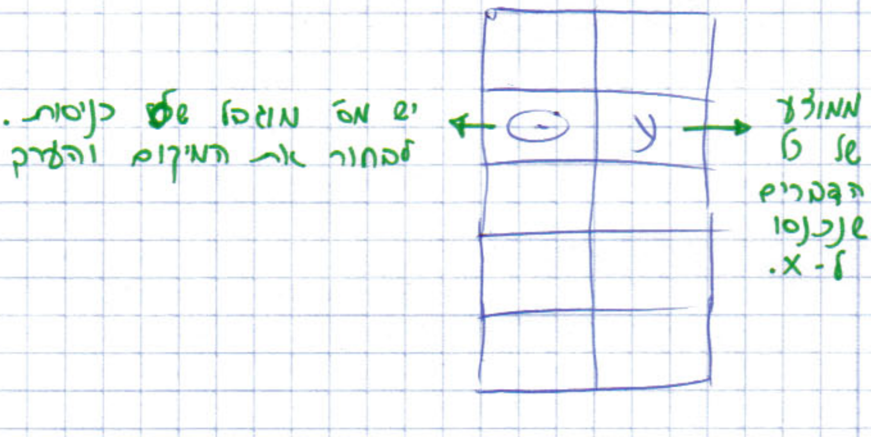
$$y_i = \begin{cases} 1 & \text{if neuron } j \text{ is the winning neuron} \\ 0 & \text{otherwise} \end{cases}$$

(*) בהתחלה נבחר הרבה שכנים להזזה כדי שיסתדרו.
לאט, לאט נצמצם כמה שכנים, כדי "לכסות" את כל ה-data points
בצורה טובה.

Counter Propogation

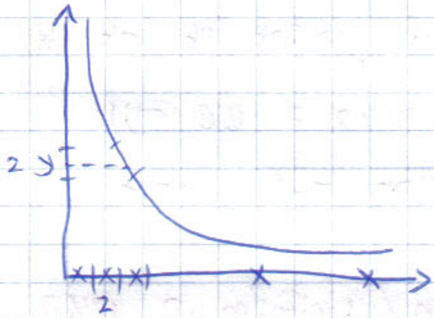
כשרוצים לבנות טכניקה של טבלה.

optimal look up table



$$\vec{y}^{\,new} := \vec{y}^{\,old} + \beta\left(\vec{y}\,' - \vec{y}^{\,old}\right)$$

$$y = \frac{1}{x}$$



יותר מהיר מ- back Propog.
אבל פחות מדויק ממנו

* ניתן להסתכל בספר Ritter להסבר טוב יותר.

full hack propogation
(שם ה-y הוא קטן)

25/7/08

נוירו-חישוביים

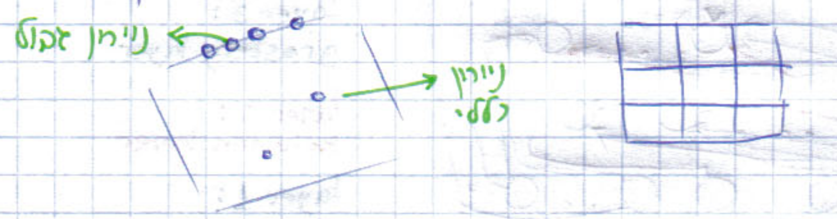
(*) kohonen נותן Equi Probable Map.

ההסתברות ש-data point "יפול" באחד מהנוירונים שווה לכל אחד מהנוירונים.

(*) הכיסוי אינו לפי ה-domain.

⇐ אפשר לבחור Sampling כך שיהיו יותר נקודות בגבול.

או טופולוגיה או מרחב המשקלים



(*) חלק מהזמן מציגים נוירון כללי ואת שכניו.

חלק מהזמן בוחרים נוירון גבולי → מתנוון לחד מימדי
ורשת טופולוגית רק על הגבול

בעיית הסוכן הנוסע

טופולוגיה: מס' נוירונים כמס' הערים, מעגל.

⇐ נימנע ממצב של:

האלגוריתם מקבל טופולוגיה ראשונית מעגל ונק' רנדומליות מייצגות את הערים.

האלגוריתם לומד את הטופולוגיה ומחבר בין הערים בצורת הטופולוגיה.

שימושים של kohonen

(*) חלק ממערכת יותר גדולה → לומדים את ההתנהגות הטופולוגית של ה-data.

(*) לומדים את מקומות ה-Clustering ואת הערכים מהתוצאות.

↳ Counter propog.

Counter Propogation

(*) מוטיבציה

$f(\vec{x}) = \vec{y}$

$\langle \vec{x}', \vec{y}' \rangle$
⋮
$\langle \vec{x}'', \vec{y}'' \rangle$

← ייצור אב טיפוסים מתאימים ל-Data ומציאות ערכיהם.

יש לנו את ה-data, ומה-Sampling
אנו רוצים לקבל את האב-טיפוסים
(אנו יודעים מראש כמה אב-טיפוסים יש לנו)
כמו כן, אנו רוצים ללמוד את הערכים שלהם
ואת אלו שקרובים להם.

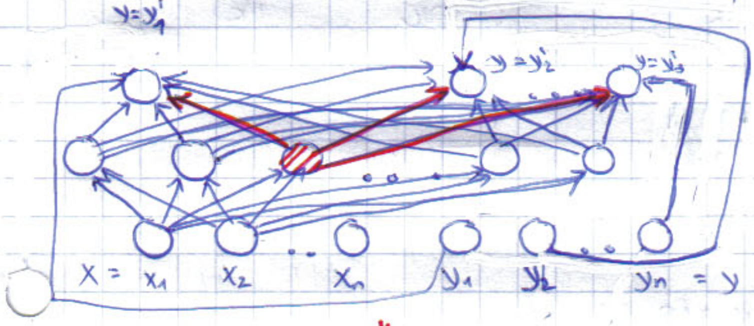
← לימוד המקומות של האב-טיפוסים - kohonen

← לימוד הערכים שלהם - Counter propogation
(נוסחה בידיעה)

(אב טיפוסים)

$$y^{new} := y^{old} + \alpha(\ldots)$$

(*) תוך כדי לימוד מעדכנים את המקום ואת הערך.



layer 3 :
grosberg layer

layer 2 :
kohonen layer

layer 1 :
input units

⇓

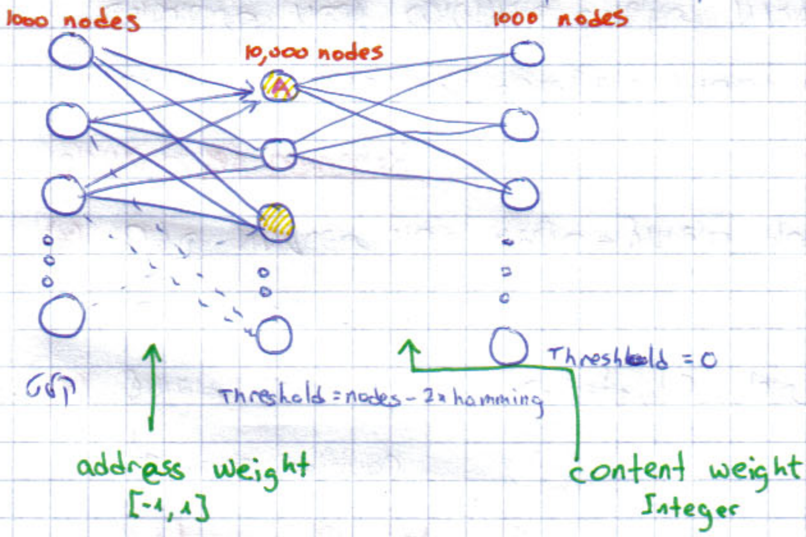
optimum look up table טוב עבור ביצוע

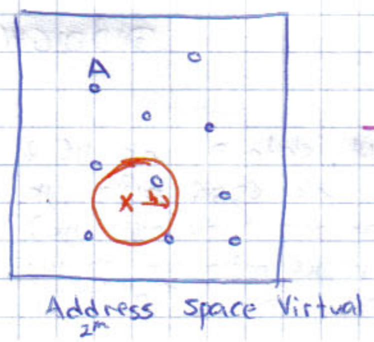
* cluster points של x (- לימוד המקומות של האב-טיפוסים)

SDM (kanerva)

(*) הרעיון - שימוש בקבוצות של נוירונים כדי לייצג את מרחב הכתובות.

יש m ביטים שמייצגים מקום בזיכרון ← 2^m כתובות.





→ כל נקודה במרחב של [-1,1]

* hamming - נמצא את כל הנקודות שמספיק קרובות (מרחק קבוע) והם יהיו מודלקים

בודקים מי מספיק קרוב (במרחק hamming) לכתובת שניתנה.

Address | 1 1 0 1 0 0 0 0 |

Address (נבחרו רנדומלית)	Content	hamming dist.	selected
00000010		4	–
00010001	~~1~~ 5 3 ~~-6~~ -7 -27 -265	3	X
01000100	~~3~~ 4 2 ~~-4~~ -5 5 2 4 4 4	3	X
01111010		4	–
10101111		7	–
10001001		4	–
10010011	~~0~~ 1 0000000	3	X
11011011	2 2 2 2 -2 -2 -2 -2	3	X

אם השפלה את התוצאה כיוון ש- 00000000

קומבינציה לינארית

Σ

| 9 7 -8 5 7 0 8 7 |

| 1 1 0 1 1 * 1 1 |

[* נניח והמרחק הינו 3.]

יכול להיות שבמקום תמונה של סבתא נקבל תמונה של סבא (כלומר, בכתובת של סבתא "נשמרה" תמונה של סבא). ← ה-content בכתובת

חזרה למבחן

Input → Computation → Output

יתרונות של NN:

- יוניברסלי
- יוניפורמי } ← ציבנקו: כל סוג רציפה יוניפורמית
- לימוד עצמי
- אלגוריתמים מקביליים
- מסביר תופעות ביולוגיות ← האלגוריתם של kohonen
- מודלים לא-לינאריים ← בגלל הסף.

חסרונות של NN:

- הלימוד הינו פרמיטיבי ← איטי למשל ב- B.P.
- המצב ההתחלתי אינו סגור ← נדרש לבצע הרבה נסיונות

לא ידוע איזה מודל מתאים לאיזו בעיה

לא ידוע איזה ארכיטקטורה טובה?

פרמטרים לא ידועים - α, מומנטום..

בעיית over training.

נדרש הרבה כוח חישוב

Non Recurrent ← ללא חוזר

perceptron - מודל פשוט, לימוד פשוט - פיתרון מובטח, מס' תרות מוגבל. ← שאלה: התכנסות של perceptron

Feed forword - נוירונים לא-לינאריים, B.P אלגוריתם

ReCurrent ← עם לולאות

Hopfield → אוטו
BAM → הטרו

- זיכרון אסוציאטיבי, קשה לשלוט שיהיו attractor שלא רוצים. אם שומרים תנאים מסוי"מים אז יש התכנסות (אם סימטרי) ← פונ' אנרגיה

לצורך הוכחה Energy / משפט התכנסות של hopefield

Kohonen - SONN, self orginizing, פשוט (לא מספיק מובן)

SDM - זיכרון אסוציאטיבי, fault tollerance

* דוגמאות של קוד יש לכל אלגוריתם.

* לא יהיה SDM ו-ART-1 (אולי תהיה שאלה על מה זה SDM)

* דוגמא לשאלה: איזה רשת מתאים על מנת לזהות רכב? נדרש להשוות בין אלגוריתמים (חסרונות ויתרונות)

* באתר קובץ Summary.ps