למידה עמוקה

סיכום קורס שהתקיים בבר-אילן בשנת 2014 ע"י פרופ' יעקב גולדברגר. סוכם והוקלד ע"י עמית מנדלבום,
Disclamier: יש כמה נושאים חשובים שחסרים כאן (Auto encoders, CNN (convent), RNN)
אך למי שאין רקע בתחום הסיכום מכסה את כל הנושאים הבסיסיים וכן קצת מעבר.

תוכן

למידה עמוקה

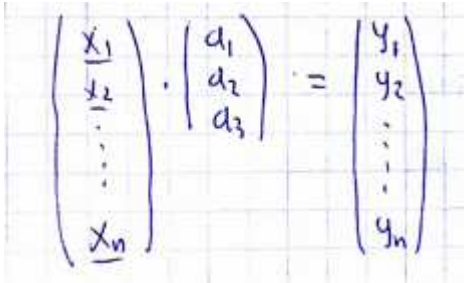
הרצאה 1

למידה עמוקה הוא תחום מתוך למידת מכונה.

דוגמא: נתון $x=(x_1,x_2,x_3)$, $y\in\{0,1\}$

אנו מחפשים מכונה לינארית $\hat{y}=a_1x_1+a_2x_2+a_3x_3=\vec{a}\vec{x}$

נגדיר פונקציית מחיר: $S(\vec{a})=\sum(\hat{y}-y)^2$ נרצה להביא למינימום. במקרה כזה ישנה פונקציה סגורה כיוון שצמצנו את עצמנו למודל לינארי:



עבור סיווג בינארי נבחר את המשערך $\hat{y}=sign(\vec{a}\vec{x})$ כלומר נקבל קו החלטה לינארי.

במקרה הלינארי קל למצוא פתרון, מציאת המשערך הלינארי הטוב ביותר היא פשוטה. הבעיה היא במישור הלא-לינארי שבו נרצה לעבוד, הוא ענק ויהיה קשה למצוא את המסווג (המכונה) הטוב ביותר.

טענה: כל פונקציה לא-לינארית ניתנת לקירוב ע"י רשת נוירונים. Logistic neural network.

הדרכים למציאת המסווג הטוב ביותר:

שנת 1986: Back propogation

שנת 1992: הומצא מודל יותר פשוט ה-SVM ונזנח ה- Neural Network

שנת 2006: חזרה ל Neural network והנושא קיבל שם חדש Deep learning.

החל מ2009 התברר כי deep learning מוכיחה את עצמה יותר משיטות אחרות.

הסיבות לרלוונטיות המחודשת של למידה עמוקה:

1) שיפור כוח החישוב בצורה דרמטית
2) גידול דרמטי במידע האימון (Training Data)

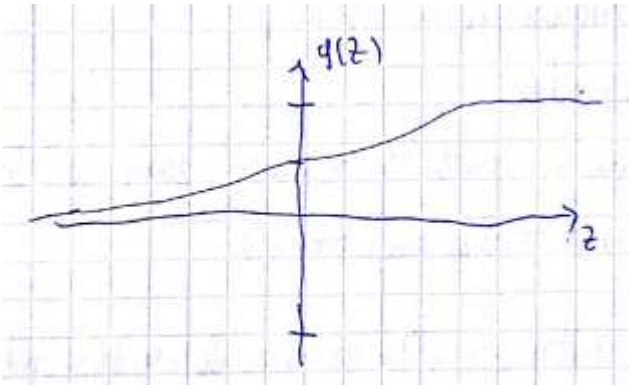
Single neuron/ logistic regression

כניסת מידע (Training data): $X_1,....X_n\in R^d$

מוצא (Labels) : $y_1....y_n\in\{0,1\}$

נחליט כי המסווג יסומן $\hat{y}$ כך ש מתקיים $\hat{y}=\begin{cases}1 & \vec{a}\vec{x}>0\\ 0 & \vec{a}\vec{x}<0\end{cases}$ ונחפש את המסווג הטוב ביותר. הבעיה שלנו עם מסווג זה היא אי-הרציפות שלו סביב 0. לכן נבצע "החלקה" של פונקצייה המטרה ע"י למשל

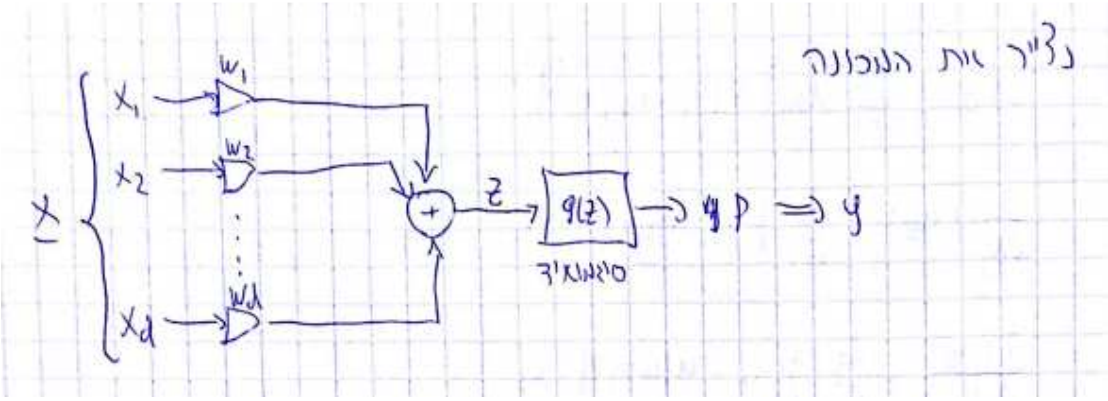
Sigmoid: $\lim_{z\to-\infty} g(z)=0 \quad \lim_{z\to\infty} g(z)=1 \qquad sigmoid(z)=\frac{1}{1+e^{-z}}=g(z),$



כעת נבנה מסווג כך שבהינתן הוקטור X נכפיל במשקלים W ונכניס לסיגמואיד כך שייתקבל:

$$z = w_1x_1 + \dots . w_d x_d$$

$$P(y=1|x)=g(z)=\frac{1}{1+e^{-z}}=\frac{1}{1+e^{-\vec{w}\vec{x}}} \qquad P(y=0|x)=1-g(z)=\frac{e^{-\vec{w}\vec{x}}}{1+e^{-\vec{w}\vec{x}}}$$



נגדיר $L(\vec{w})=\sum_t \log\{p(y_t \mid x_t, w_t)\}$ ונחפש W כך L מקסימלי כלומר $\hat{w}=\arg\max\{L(\vec{w})\}$.

לאחר שנמצא W אופטימלי נקבל מכונה שמוציאה soft-desicion על סמך X חדש (לא ידוע)

נבצע כמה חישובי עזר:

$$\frac{d}{dz}\log[g(z)]=1-g(z) \qquad \frac{d}{dz}\log[1-g(z)]=-g(z)$$

$$\frac{\partial}{\partial w_i}\{\log[P(y=1|x,w)]\}=\frac{\partial}{\partial w_i}\{\log[g(wx)]\}=[1-g(wx)]x_i$$

$$\frac{\partial}{\partial w_i}\{\log[P(y=0|x,w)]\}=\frac{\partial}{\partial w_i}\{1-\log[g(wx)]\}=-g(wx)x_i$$

כך שלסיכום נקבל:

$$\vec{w},\vec{x}\in R^d, \qquad \vec{w}\vec{x}\in R, \qquad g(\vec{w}\vec{x})\in R$$

$$L(\vec{w})=\sum_t \log\{p(y_t\mid \vec{x_t},\vec{w})\} \qquad \frac{\partial}{\partial w_j}L(\vec{w})=\sum_t [y_t-g(\vec{x_t}\vec{w})]x_j$$

$$\frac{\partial L(\vec{w})}{d\vec{w}}=\sum_t [y_t-g(\vec{x_t}\vec{w})]\vec{x_t}=\sum_t [y_t-p(y_t=1\mid \vec{x_t},\vec{w})]\vec{x_t}=\sum_t [y_t-\hat{y}_t]\vec{x_t}$$

נשים לב כי $\frac{\partial L(\vec{w})}{d\vec{w}}$ הוא וקטור.

כעת כדי למצוא את w האופטימלי (כלומר להביא את הנגזרת ל-0) נשתמש בשיטת **Gradient Ascent:**

$\vec{w}_{t+1}=\vec{w}_t+\varepsilon\frac{\partial}{\partial\vec{w}}L(\vec{w}_t)=\vec{w}_t+\varepsilon\sum_t[y_t-\hat{y}_t]\vec{x_t}$ כאשר ε נקרא קצב הלמידה (**learning rate**)

ניתן להוכיח (יש בדפים) כי L(w) הינה פונקציה קעורה ולכן עם מקסימום בודד.

בסיום התהליך נקבל w אופטימלי ואז קו החלטה לינארי כמו שראינו קודם.

<u>הרצאה 2</u>

Multi-Class logistic regression

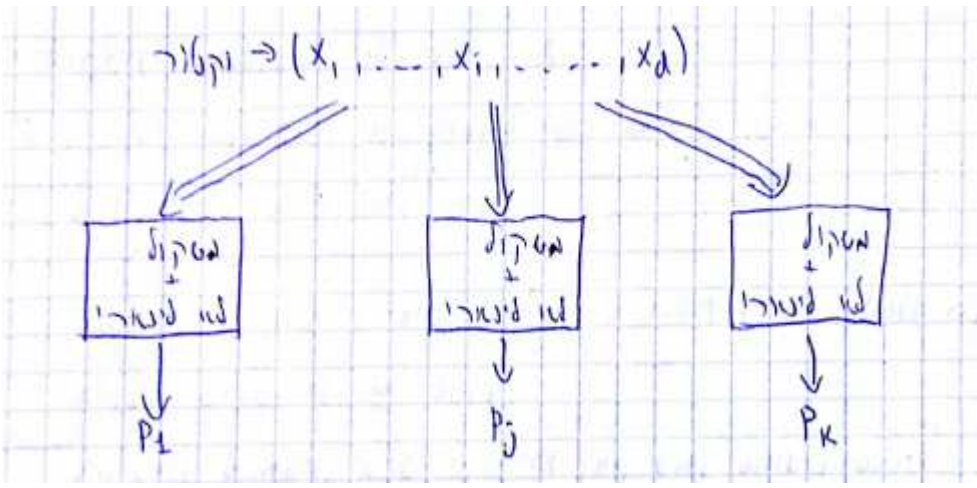
במקרה הזה ל-y יש יותר משתי אפשרויות {0,1} :

$$Input: x_1...x_n\in R^d \qquad Output: y_1..y_n\in\{1,...k\}$$

ואז: $P(y=k\mid x)=\frac{e^{\vec{w}_i\vec{x}}}{\sum_j e^{\vec{w}_j\vec{x}}}=soft\max(w_1x,w_2x...w_kx)$

זאת לעומת : $\arg\max\{p(y=i\mid x)\}=\arg\max\{w_ix\}$

המודל הגרפי של המערכת:



תהליך החלטה:

1) לוקחים X עם d רכיבים
2) מכפילים את הוקטור x במשקלים w_j ומקבלים k ערכים

3) מכניסים את k הערכים ל $p(y=i\mid x)$ שראינו קודם, ונקבל התפלגות על k ערכים.
4) מחליטים על הערך של y לפי המקסימום מביניהם.

נגדיר פונקציית מחיר $S(w)=\sum_t \log[p(y_t\mid x_t,w)]$ שוב נחפש w אופטימלי (כלומר שמביא פונקציה זו למקסימום), כלומר נגזור את ההסתברות לקבל j לפי הוקטור w_i:

:

$$\frac{\partial}{\partial \vec{w}_i}\log\left(\frac{e^{\vec{w}_j\vec{x}}}{\sum_j e^{\vec{w}_j\vec{x}}}\right)=\frac{\partial}{\partial \vec{w}_i}[\vec{w}_j\vec{x}-\log(\sum_j e^{\vec{w}_j\vec{x}})]=\delta_{ij}\vec{x}-\frac{e^{\vec{w}_j\vec{x}}}{\sum_j e^{\vec{w}_j\vec{x}}}\vec{x}$$

$$=\left(\delta_{ij}-\frac{e^{\vec{w}_j\vec{x}}}{\sum_j e^{\vec{w}_j\vec{x}}}\right)\vec{x}=[\delta_{ij}-p(y=i\mid x,w)]\vec{x}$$

כאשר δ_{ij} הוא 1 רק אם i=j.

כעת באותה דרך נגזור את פונקצית המחיר ונקבל:

$\frac{\partial S}{\partial \vec{w}_i}=\sum_t[1_{\{y_t=i\}}-P(y_t=1\mid x_t)]\vec{x}_t$ כאשר $1_{\{y=i\}}$ הוא 1 רק אם y=i ואפס במקום אחר (פונקציית אינדיקטור). גם כאן ניתן להוכיח כי קיים מקסימום גלובלי אחד.
וכעת באותה דרך נבצע Gradient Ascent רק שעכשיו יש לבצע K פעמים עבור כל אחד מווקטורי ה-W.

הערה: במקרה הבינארי ראינו כי ניתן להשתמש בוקטור W אחד. נראה איך זה יוצא מכאן:

$$P(y=1\mid x)=\frac{e^{w_1x}}{e^{w_1x}+e^{w_0x}}=\frac{1}{1+e^{-(w_1-w_0)x}}=sigmoid[(w_1-w_0)x]$$

<u>מקרה רציף</u>

במקרה הרציף נשתמש בשיטה אחרת: **Linear Regression** (בניגוד ל Logistic regression שראינו עד כה):

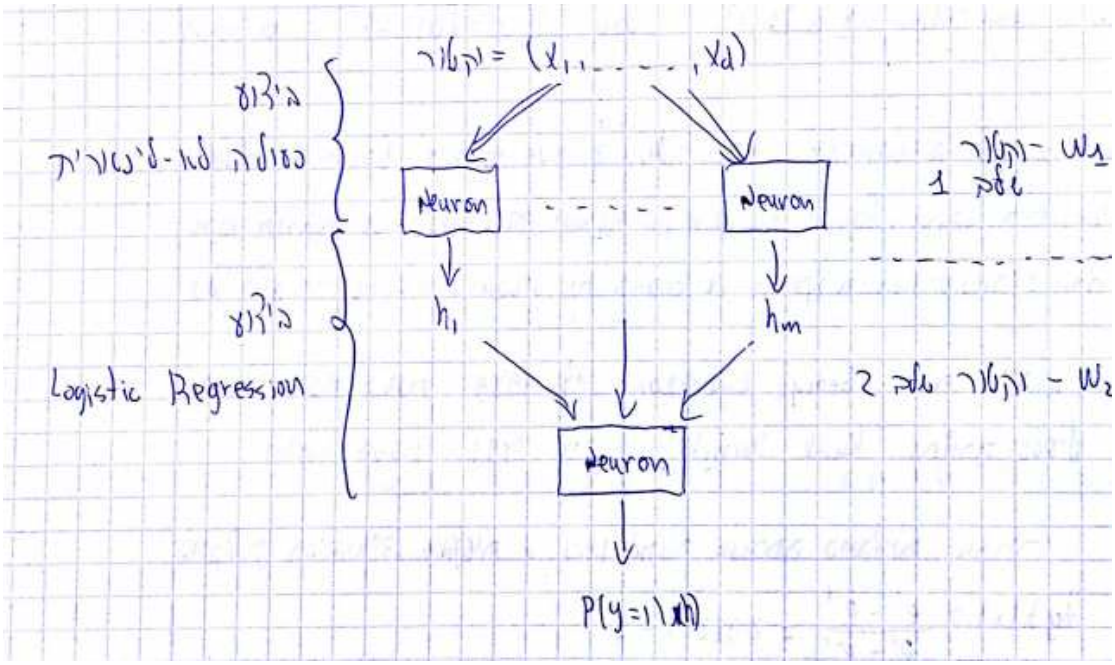
$$Input: x_1...x_n\in R^d \qquad ouput: y_1...y_n\in R$$

$$\hat{y}=\vec{w}\vec{x} \qquad S(\vec{w})=\sum_t(\vec{w}\vec{x}_t-y_t)^2$$

קיימת כאן הנחה מובלעת כי y מתפלג נורמלית עם תוחלת wx.

Neural network with 1 hidden layer

נשים לב כי המסווג שראינו לעיל הוא לינארי וכמו שאמרנו זוהי קבוצה קטנה של מסווגים. אנו רוצים לחזור לעולם הלא לינארי. הדרך לעשות זאת היא להעביר את X דרך פעולה לא-לינארית ואז להשתמש במסווג לינארי כמו שראינו למעלה, אם נחזור למקרה הבינארי הרשת תיראה כך:



כך ש: $P(y=1|x)=\frac{1}{1+e^{-\vec{w}_2\vec{h}}}$ $\quad h_i=\frac{1}{1+e^{-\vec{w}_{1i}\vec{x}}}$ כאשר w_1 קבוצת הווקטורים בשלב הראשון

הדבר הזה נקרא **רשת נוירונים**

<u>מודל רשת הנוירונים</u>

1) שלב הפעולה הלא-לינארית בו משתמשים ב- LR אך לא כמסווג
2) שלב הסיווג בו ניתן להשתמש בכל מסווג (בינארי, Multiclass וכו')

<u>משפט הקירוב האוניברסאלי</u>

תהי $\varphi(x)$ פונקציה רציפה מונוטונית עולה וחסומה לדוגמא סיגמואיד. ותהי f(x) פונ' רציפה אז לכל ε קיים $\hat{f}(x)=\sum_i a_i\varphi(w_i x+b_i)$ כך ש $|f(x)-\hat{f}(x)|<\varepsilon$

במילים אחרות כיוון שהשלב הראשון ברשת הנוירונים מורכב מסיגומואידים אזי הניורון האחרון הוא $\hat{f}$, ומכאן שבעזרת רשת נוירונים ניתן ליצור קירוב לכל פונקציה שנרצה.

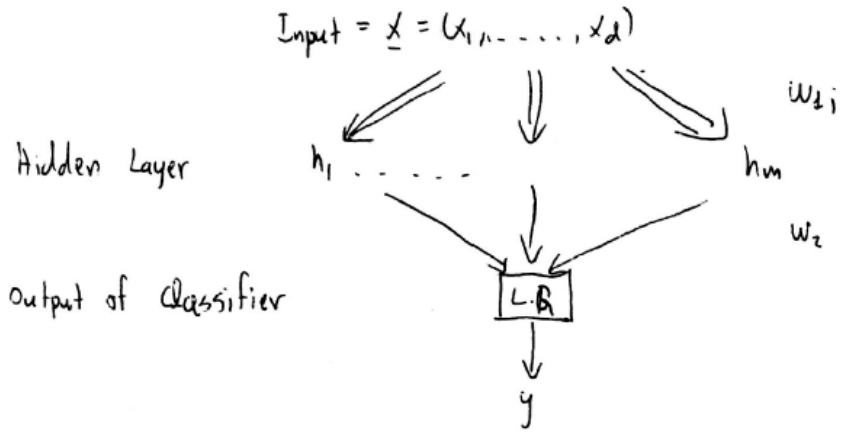
כעת יש צורך לאמן את המודל. נגדיר $L(w)=\sum_t \log P(y_t\,|\,\vec{x}_t,\vec{w})$

הבעיה היא שכעת יש לנו הרבה נקודות מקסימום מקומיות למראות זאת עדיין נשתמש ב – Gradient Ascent, כלומר $w_{t+1}=w_t+\varepsilon\frac{\partial L}{\partial w}(w_t)$

הבעיה היא שצריך לעבור על כל הרשת כל פעם עבור כל אחד מהפרמטרים (כל איבר בכל וקטור w) לכן המציאו שיטה מהירה יותר שעושה הכל במכה (כלומר גוזרת לפי כל ה-w בכל מעבר על הרשת)

הרצאה 3

תזכורת ברשת נוירונים עם שכבה נסתרת אחת:



$$h_i = \frac{1}{1+e^{-(\vec{w}_{1i}\vec{x}+b_{1i})}} \qquad p(y=1|h) = \frac{1}{1+e^{-(\vec{w}_2\vec{h}+b_2)}}$$

כעת שוב נחפש w ו-b המוצאים מקסימום לפונקציית L בעזרת Gradient ascent:

$$L(w,b) = \sum_t \log[P(y_t \mid \vec{x}_t, \vec{w}, b)]$$

$$w_{t+1} = w_t + \varepsilon \frac{\partial S}{\partial \vec{w}}(w_t) \qquad b_{t+1} = b_t + \varepsilon \frac{\partial S}{\partial b}(b_t)$$

כעת כמו שהזכרנו, זה לא יעיל לגזור עבור כל איבר בכל אחד מווקטורי ה w בנפרד לכן נלמד שיטה:

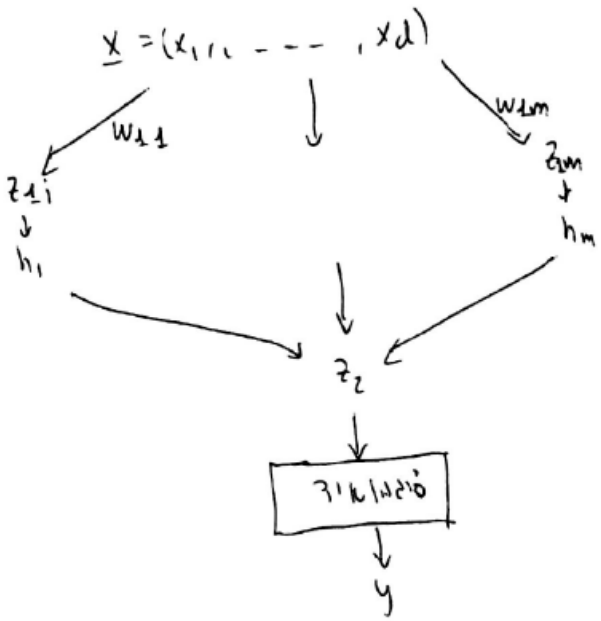
Back propagation algorithm

Feed forward

נעבור על הרשת מלמעלה למטה כרגיל כך ש:

$$z_{1i} = \vec{w}_{1i}\vec{x} + b_1 \qquad z_2 = \vec{w}_2\vec{h} + b_2$$

$$h_i = g(z_{1i}) \qquad \hat{y} = p(y=1|x,w,b) = g(z_2)$$



Back propagation

כפי שראינו בהרצאה הראשונה: $\frac{\partial S}{\partial z_2} = (y - \hat{y})$ ולכן

$$\frac{\partial S}{\partial w_2} = \frac{\partial S}{\partial z}\frac{\partial z_2}{\partial w_2} = (y - \hat{y})h = [y - p(y = 1 \mid x, w, b)]h$$

$$\frac{\partial S}{\partial b_2} = \frac{\partial S}{\partial z}\frac{\partial z_2}{\partial b_2} = (y - \hat{y})$$

כעת עבור השלב הראשון:

$$\frac{\partial S}{\partial w_{1i}} = \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial h_i}\frac{\partial h_i}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial w_{1i}} = (y - \hat{y})w_{2i}g'(z_{1i})\vec{x} = (y - \hat{y})w_{2i}h_i(1 - h_i)\vec{x}$$

$$\frac{\partial S}{\partial b_{1i}} = \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial h_i}\frac{\partial h_i}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial b_{1i}} = (y - \hat{y})w_{2i}h_i(1 - h_i)$$

Non-linear regression – <u>דוגמא נוספת</u>

$Input: x_1 ... x_n \in R^d \qquad Output: y_1 .... y_n \in R$

Feed forward

$$h_i = \frac{1}{1 + e^{-(\vec{w}_{1i}x + b_{1i})}} \qquad \hat{y} = \vec{w}_2\vec{h} + b_2$$

נגדיר: $L(w,b) = \sum_t (y_t - \hat{y}_t)^2 \qquad S = (y_t - \hat{y}_t)^2$

Back propagation

$$\frac{\partial S}{\partial \hat{y}} = -2(y-\hat{y})h = 2(\hat{y}-y) \qquad \frac{\partial S}{\partial w_2} = \frac{\partial S}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w_2} = 2(\hat{y}-y)h \qquad \frac{\partial S}{\partial b_2} = \frac{\partial S}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial b_2} = 2(\hat{y}-y)$$

$$\frac{\partial S}{\partial h} = \frac{\partial S}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h} = 2(y-\hat{y})w_2 \qquad \frac{\partial S}{\partial z_{1i}} = \frac{\partial S}{\partial h_i}\frac{\partial h_i}{\partial z_{1i}} = [2(y-\hat{y})w_2]g'(z_{1i}) = [2(y-\hat{y})w_2]h_i(1-h_i)$$

$$\frac{\partial S}{\partial w_{1i}} = \frac{\partial S}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial w_{1i}} = [2(y-\hat{y})w_2]h_i(1-h_i)\vec{x}$$

$$\frac{\partial S}{\partial b_{1i}} = \frac{\partial S}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial b_{1i}} = [2(y-\hat{y})w_2]h_i(1-h_i)$$

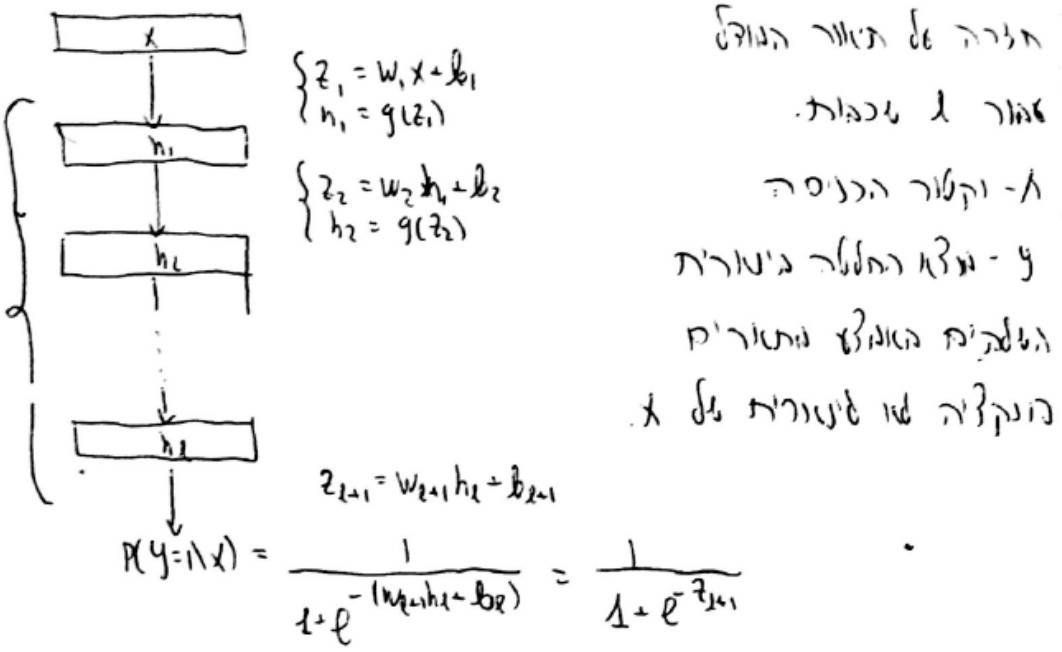
תהליך ה- Feed forward אינו בהכרח לינארי

תהליך ה Back propogation הינו תמיד לינארי

<u>הרצאה 4</u>

Deep Neural network

כשאנו מדברים על DNN הכוונה היא לרשתות נוירונים עם יותר מ-hidden layer אחד:



גם כאן משתמשים בדיוק באותו FF ו –BP שראינו ב- Single layer אך כעת נשים לב כי W בנגזרות הינו וקטור ולא סקלר כמו מקודם. כמו כן נזכיר כי h_L ו w_{L+1} הינם בעלי אותו מימד

ראה כל הנוסחאות בדפים המצורפים להרצאה 3

אתחול פרמטרים

עד כאן דיברנו רק על הדרך המהירה למצוא W אופטימלי, כעת נדבר על אתחול.

תחילה נשים לב שברשת נוירונים רגילה אם משקולות מסוימים יהיו זהים גם המוצא יהיה זהה, לכן נרצה להגריל תנאי התחלה כך שתאים שונים יקבלו אתחול שונים.
כמו כן יש לעדכן את המשקלים כך שערכי הסיגמואיד ייפלו באיזור הלינארי של הסיגמואיד (בו הנגזרת גבוהה יחסית) ולכן נרצה להגריל באיזור הרלוונטי (קרוב יחסית ל-0) **כך ש**:

$w \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ כאשר n מספר האבירים בווקטור x, וכן $b = 0$ (אם כי גם אותו מומלץ להגריל אקראית בין 0 ל-1)

מאותה סיבה **נרצה לנרמל את X** כך שגם יהיה קטן (למשל אם כל איבר בוקטור X מקבל ערכים בין 0 ל-1 אזי נחלק את X ב- n).

אופטימיזציה

נגדיר פונ' חדשה: **Cross Entropy**: $S(\vec{w}) = -\frac{1}{n}\sum_i \log[p(y_i \mid x_i, w)]$

נגדיר $S_i(\vec{w}) = -\log[p(y_i \mid x_i, w)]$ כך ש $S(\vec{w}) = \frac{1}{n}\sum_i S_i$

את הפונקציה הזו נרצה להביא למינימום לכן נעבוד עם **Gradient Descent:**

$$w_t = w_{t-1} - \varepsilon \frac{\partial s}{\partial w}(w_{t-1})$$

כעת נגדיר **SGD (Stochastic gradient descent):**

$w_t = w_{t-1} - \varepsilon \frac{\partial s_i}{\partial w}(w_{t-1})$ כלומר בכל צעד נחשב לפי נקודה אחת ב- Data ולא לפי כל המידע הדבר הזה יותר מהיר מאשר לעבור על כל הטור כל פעם .

נגדיר שיטות בהקשר הזה:
Full batch: גוזרים כל פעם לפי כי מידע הלמידה
Online: גוזרים כל פעם רק לפי יחידת מידע אחת (אקראית)
Mini batch: מסדרים את המידע בקבוצות בגודל M ואז עבור כל קבוצה M מבצעים Full batch.
הערה: שיטת Online היא Mini batch עבור M=1. ושיטת Full batch הינה שיטת Mini batch עבור M=n.
הערה נוספת: כאשר עובדים עם Mini batch חשוב "לערבב" את המידע לפי שמחלקים לקבוצות כך שלא נקבל בקבוצה מסוימת תווית אחת (כלומר מידע מסוג מסוים אחד) אלא נקבל מכל התוויות.

<u>Learning rate</u>
אין חוקים ברורים פה, עושים לפי מה שעובד. ניתן אולי רק לומר שככל שמתקרבים למקסימום\מינימום אפשר להקטין אותו

Momentum method

בשיטת GA או GD אנחנו זזים בכיוון הגרדיאנט אך ייתכן שהגרדיאנט לכיוון המינימום מקסימום הוא קטן ולכן ייקח לנו זמן להתכנס אליו. שיטה זו אמורה לעזור לנו.

נגדיר $\Delta_t = -\varepsilon \frac{\partial s}{\partial w}(w_{t-1}) + \alpha \Delta_{t-1}$ כאשר $\alpha \sim [0,1]$ נקרא <u>מקדם המומנטום</u> הדבר הזה מאפשר לנו למצוא את "הכיוון הנכון" של הגרדיאנט יותר בקלות זאת כי כיוון שאנו מצרפים את הנגזרות מזמנים קודמים. כך שהגרדיאנט בכיוון הנכון, גם אם הוא קטן, הוא קבוע ולכן מצטבר, ואילו הגרדיאנטים הלא נכונים אולי גדולים אך בכיוונים משתנים ולכן מתבטלים עם הזמן (אם מצרפים אותם).

כלומר נקבל $w_t = w_{t-1} - \varepsilon \frac{\partial s}{\partial w}(w_{t-1}) + \alpha \Delta_{t-1}$

הדבר הזה כמובן עוזר לאוסלציות אך עדיין לא יימנע מאתנו להיכנס למינימום מקומי.

<u>Nesteror momentum method</u>

שינוי קל לשיטה הקודמת כך ש: $w_t = w_{t-1} + \alpha \Delta_{t-1} - \varepsilon \frac{\partial s}{\partial w}(w_{t-1} + \alpha \Delta_{t-1})$

הדבר נובע מכך שאני יודע שבכל מקרה לפי שיטת המומנטום אני ממילא זז לכיוון $w_{t-1} + \alpha \Delta_{t-1}$ לכן אפשר כבר מראש לעשות את הנגזרת שם.

<u>הרצאה 5</u>

Dropout

הרעיון כאן הוא שבזמן הלמידה אנחנו מאפסים באופן אקראי חלק מהניורונים (חלק מה-h)

ניתן טיפה רקע:

נניח כי נתונים לנו מספר מסווגים כך ש:

$P_{j1} = P(y=1|x)$ משתמש במסווג j.

אפשרויות למיצוע:

מיצוע חשבוני: $P(y=1|x) = \frac{1}{k}\sum_j P_{j1}$

מיצוע גאומטרי: $\frac{P(y=1|x)}{P(y=0|x)} = \left(\prod_j \frac{P_{j1}}{P_{j0}}\right)^{\frac{1}{k}}$ או בכתיב לוגריתמי $\frac{1}{k}\sum_j \log\left(\frac{P_{j1}}{P_{j0}}\right)$

נגיד שיש לנו m כמות של h , אזי כל אחד מהם מקבל ערך של 0 או 1 כך שמקבלים 2^m סוגים של רשתות (בכל אחת מהרשתות מאפסים h אחרים) וכל פעם מאמנים אחת מתוך הרשתות האלה מה שייקרה בסוף הוא שנקבל 2^m מסווגים ובסופו של דבר בזמן הTest פשוט נעשה מיצוע

$$P_T(y=1|x) = \frac{1}{1+e^{-(\frac{1}{2}w_2 h + b_2)}} \qquad P_A(y=1|x) = \frac{1}{1+e^{-(\sum w_{2i} h_i a_i + b_2)}}$$

כלומר בזמן הלמידה כל פעם מאפסים h אחר, כלומר מכפילים במשתנה אקראי a שהינו 1 או 0 עבור כל אחד מה-h ובזמן הTest פשוט עושים מיצוע על ידי הכפלה בחצי.

נשים לב כי בשיטה זו כאשר עושים Back propagation אין צורך לבצע גזירה עבור המשקלים של ה- h שאותם איפסנו (הנגזרת תהיה שווה לאפס פשוט), נשים לב כי מדובר הן על המשקלים שהולכים אל אותו ה- h וכן על המשקלים שיוצאים ממנו לעבר y.
(**ראה דפים של dropout לנוסחאות המעודכנות**)

(ראה הוכחה לנכונות ה- Dropout בדפים של אלון הרצאה חמש דף אחרון)
סיכום: בכל מעבר על הרשת (בין אם עבור כל מידע או mini-batch וכו') נאפס חלק מהנוירונים כך שתתקבל תת רשת אקראית. נמשיך כך עד שנעבור על כל האפשרויות של הרשתות האקראיות (ייתכן שיידרשו כמה שימושים בכל המידע בשביל זה) .
בסיום נקבל סט פרמטרים להם ניתן פקטור 1/2 ובהם נשתמש בזמן ה-Test
הדבר הזה שקול לאימון על 2^m רשתות שונות

<u>הרצאה 6</u>

RBM – Restricted Boltzman machine

אלגוריתם זה קשור לנושא ה- Unsupervised pre-training , מטרתו היא למצוא אתחול בצורה מושכלת לפרמטרים.
משמעות ה- Unsupervised היא שאנחנו משתמשים ב- Data בלבד ללא Labels. בדר"כ המידע זמין יותר לעומת labels שיותר קשה למצוא. נניח $\vec{x} \in \{0,1\}^n$
נתבונן בפונקציית אנרגיה $E : \vec{x} \to \mathbb{R}$, אם נמיר למודת הסתברותי נקבל:
$P(\vec{x}) = P(x_1....x_n) = \frac{e^{-E(\vec{x})}}{\sum_{\vec{x}} e^{-E(\vec{x})}}$ המכנה מהווה גורם נרמול ונקרא Partition factor

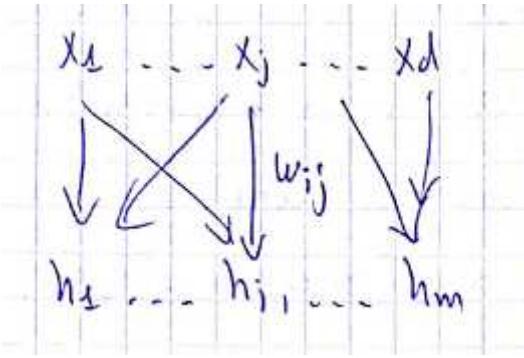
למשל דוגמא. נניח X מתפלג אחיד, ונניח $E(x_1,...x_n) = 7$ אז $P(\vec{x}) = P(x_1....x_n) = \frac{e^{-7}}{\sum_{\vec{x}} e^{-7}} = \frac{1}{2^n}$

המכנה נובע מכך שיש 2^n אפשרויות עבור הוקטור x. כלומר לפונקציית אנרגיה יש 2^n התאמות, אחת עבור כל אפשרות של X.

נתבונן על פונקציית אנרגיה ריבועית: $E(\vec{x}) = \sum_{ij} w_{ij} x_i x_j - \sum_i b_i x_i$ הדבר הזה נקרא הסתברות בולצמן (**Boltzman machine**) כעת נחלק את הוקטור X לשתי קבוצות: $\vec{x} = (\vec{x}, \vec{h})$
וייתקבל:

$$E(\vec{x}) = E(\vec{x}, \vec{h}) = -\sum_{ij} w_{ij} h_i x_j - \sum_j a_j x_j - \sum_i b_i h_i = -\vec{h}^T \vec{w} \vec{x} - \vec{a}\vec{x} - \vec{b}\vec{h}$$

זהו מודל **RBM**, כך ש m+d=n ו $h \in \{0,1\}^m$ $x \in \{0,1\}^d$
בצורה גרפית זה ייראה כך:



כלומר דואגים לכך שבחלוקה ל- X ו- h לא תתקבל הכפלה של x ב- x ושל h ב- h, לכן זה נקרא restricted.

כעת $$P(x,h)=\frac{e^{-E(\vec{x},\vec{h})}}{\sum_{x,h} e^{-E(\vec{x},\vec{h})}}=\frac{e^{\vec{h}^T\vec{w}\vec{x}+\vec{a}\vec{x}+\vec{b}\vec{h}}}{\sum_{x,h} e^{\vec{h}^T\vec{w}\vec{x}+\vec{a}\vec{x}+\vec{b}\vec{h}}}$$

טענה: $P(h_i=1|x)=sigmoid(\sum_j w_{ij}x_j+b_i)$ $\qquad P(h|x)=\prod_{i=1}^{m}P(h_i|x)$

הוכחה:

$P(\vec{h}|\vec{x})\overset{Bayes}{=}\frac{P(\vec{x},\vec{h})}{P(\vec{x})}=\frac{e^{-E(\vec{h},\vec{x})}}{Const}\propto e^{-E(\vec{h},\vec{x})}$ נשים לב כי הקבוע במכנה הוא בגלל ש-X נתון.

כעת $e^{\vec{h}^T\vec{w}\vec{x}+\vec{a}\vec{x}+\vec{b}\vec{h}}=\exp(\sum_i \vec{h}_i\vec{w}_i^T\vec{x}+b_ih_i)=\prod_i\exp[(\vec{h}_i\vec{w}_i^T\vec{x})+\vec{b}_i\vec{h}_i]$ כאשר את המקדם ax הורדנו כי הוא קבוע (x נתון ו-a וקטור קבוע כלשהו). והוכחנו ש $P(h|x)=\prod_{i=1}^{m}P(h_i|x)$

נשים לב כי הצלחנו לפרק את פונקצית צפיפות ההסתברות למכפלה של גורמים שונים ולכן אותם גורמים הם בת"ס, כך שעד כדי נרמול קיבלנו את ההתפלגות הבאה:

$$P(h_i|x)=\exp(\vec{h}_i\vec{w}^T\vec{x}+\vec{h}_i\vec{b}_i)$$

ואם ניזכר כי $h\in\{0,1\}$ נקבל כי:

$$P(h_i=0|x)=e^0=1$$
$$P(h_i=1|x)=e^{\vec{w}_i^T\vec{x}+b_i}$$

ונזכיר שהסתברויות אלה ללא נרמול (לכן סכומן גדול מ-1)

לאחר נרמול נקבל כי: $$\frac{P(h_i=1|x)}{P(h_i=1|x)+P(h_i=0|x)}=\frac{e^{\vec{w}_i^T\vec{x}+b_i}}{1+e^{\vec{w}_i^T\vec{x}+b_i}}=\frac{1}{1+e^{-(\vec{w}_i^T\vec{x}+b_i)}}$$

כלומר קיבלנו **סיגמואיד!** בדיוק כמו ב-Logistic regression. והוכחנו את הטענה.

לסיכום חלק זה: אם נסתכל על המודל ההסתברותי בצורה גרפית אזי בהינתן פונקציה אנרגיה של X ו-h ומודל RBM, אזי ההסתברות של h בהינתן X היא סיגמואיד, בדיוק כמו ברשת נוירונים

כעת נחשב את ההסתברות השולית של X כאשר 1/z גורם הנרמול:

$$P(\vec{x}) = \frac{1}{Z}\sum_{\vec{h}} e^{\vec{h}\vec{w}\vec{x}+\vec{a}\vec{x}+\vec{b}\vec{h}} = \frac{1}{Z}e^{a\vec{x}}\sum_{\vec{h}} e^{\vec{h}\vec{w}\vec{x}+\vec{b}\vec{h}} = \frac{1}{Z}e^{a\vec{x}}\sum_{\vec{h}} e^{\sum_i \vec{h}_i\vec{w}_i\vec{x}+b_i h_i}$$

$$= \frac{1}{Z}e^{a\vec{x}}\sum_{\vec{h}}\left(\prod_i e^{\vec{h}_i\vec{w}_i\vec{x}+b_i h_i}\right) = \frac{1}{Z}e^{a\vec{x}}\prod_i\left(\sum_{h_i=\{0,1\}} e^{\vec{h}_i\vec{w}_i\vec{x}+b_i h_i}\right) =$$

$$\frac{1}{Z}e^{a\vec{x}}\prod_i\left(\underset{h_i=0}{1}+\underset{h_i=1}{e^{\vec{w}_i\vec{x}+b_i}}\right)$$

(חסר פה את המשך ההרצאה שקשור לאימון RBM ראה 2 עמודים אחרונים ברצאה שש אצל אלון או בדפים, בגדול זה פחות רלוונטי למבחן)

<u>הרצאה 7</u>

המטרה שלנו במודל RBM היא למצוא את הפרמטרים w ו-b ולהשתמש בפרמטרים אלו לטובת Neural network. נדרש כמובן לאתחל את מודל ה- RBM

Gibbs Sampling

נרצה לדגום את המ.א. (x,h) דהיינו לדגום n דגימות מ-x ו- m דגימות מ-h כך שהם ייתפלגו לפי P(x,h). נזכיר כי במקרה שלנו $P(x,h) = \frac{1}{z}e^{-E(x,h)}$

כמו כן:

$$P(h\,|\,x) = \prod_i P(h_i\,|\,x) \to P(h_i=1\,|\,x) = g(\sum_j w_{ij}x_j + b_i)$$

$$P(x\,|\,h) = \prod_j P(x_j\,|\,h) \to P(x_j=1\,|\,h) = g(\sum_i w_{ij}h_i + a_j)$$

כעת מה שנעשה הוא שבהינתן x_1 כלשהו ראשוני (נבחר אותו אקראית) נוכל להגריל את h_1 לפי פונקצית צפיפות הפילוג המותנה ומתוך אותו h_1 להגריל את x_2 לפי פונקצית צפיפות הפילוג המותנה השנייה וכן הלאה מ-x ל-h: $(x_1 \to h)_1 \to (x_2 \to h_2) \to \ldots (x_t \to h_t)$

כך עבור t גדול מספיק $(x_t, h_t) \sim P(x,h)$.

כעת בהינתן שזורקים את הדגימות הראשונות (שאינן מתפלגות עדיין לפי פונקציית הפילוג, למשל את 100 הראשונות) אזי משם והלאה כל הדגימות שניקח ניתן להניח כי הן מתפלגות לפי P

אם נרצה למשל לחשב תוחלת $\eta[f(x,h)] = \frac{1}{T}\sum_t f(x_t,h_t)$ כאשר t גדול מספיק

או אם למשל נרצה לחשב תוחלת של איבר ספציפי $\eta(x[3]) \approx \sum_t x_t[3]$

נניח כעת שנתונים לנו n דגימות של x כך ש: $x_1,\ldots x_n \in \{0,1\}$ אזי:

$S(\vec{w},\vec{a},\vec{b}) = \frac{1}{n}\sum_t \log P(x_t; w,a,b)$ אם w,a,b ידועים ניתן למצוא התפלגות P(x,h) ומתוכה את P(x) כמו שמצאנו להלן. כעת בעזרת Gibbs sampling ניתן לדגום X –h כלומר, בהינתן מידע למידה x_t אזי $h_{t1,i} \sim p(h_i \mid x_t) \rightarrow x_{t2,j} \sim P(x_j \mid h_{t1})$

ומתוכם ה-X שהתקבלו לחשב את S , וכמובן שנרצה למקסם את S ע"י w,a,b ונעשה זאת ע"י Gradient Ascent

(ראה נגזרות ואלגוריתם בדפים המצורפים להרצאה 6)

כלומר נעשה כך:

1) נגריל w,a,x התחלתיים (כמו ברשת נוירונים)
2) נגריל h בהינתן x ואז נגריל x בהינתן h
3) נעשה נגזרות לפי הנוסחאות ולפי ה- X וה- h שמצאנו ונתקדם לפי גרדיאנט אסנט
4) נחזור לצעד 2 ונמשיך כמה שרוצים

כלומר מדובר פה על אימון של רשת RBM, לאחר שנאמן אותה נמצא למעשה w,b המאפשרים להגיע לייצוג טוב יותר של X ואותם ניקח אתנו אל רשת הנוירנים.

לסיכום: RBM הינה שיטת Pre-training בה אנו רוצים לאתחל את הפרמטרים w,b בצורה אופטימלית.

הקשר בין RBM לרשת נוירונים:

1) בהינתן וקטור X
2) מוצאים W,a,b בעזרת RBM
3) מוצאים P(x,h)
4) מוצאים $P(h_i = 1 \mid x) = g(\sum_j w_{ij} x_j + b_j)$ למעשה חזרנו לרשת נוירונים

הערה: בהינתן כמה שכבות של רשת בהינתן X נגריל h ואז h הוא ה- x של השכבה השנייה וכן הלאה.

כמו כן נזכיר כי בRBM $h \in \{0,1\}$ לעומת רשת נוירונים בה $h \in R$.

השימוש ב- RBM נועד כדי למצוא פרמטרים התחלתיים טובים יותר עבור w,a,b. הדבר שימושי במיוחד כאשר יש לנו מעט מידע ללמידה (כלומר מעט מידע עם labels)