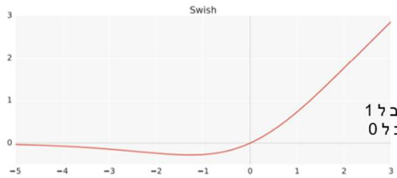
Swish Self Gated Activation

$\sigma(z)$ =z sigmoid(z)

Sigmoid(z) $=\frac{1}{1+\exp(-z)}$



Z חיובי, sigmoid(z) מתקרב ל 1
Z שלילי: sigmid(z) מתקרב ל 0

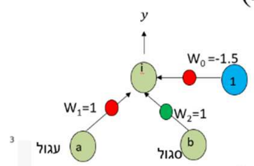
1. Spike Time Dependent Plasticity-STDP
נתונים 2 נוירונים א ו ב. א מחובר דרך סינפסה A לנוירון ב ואילו נוירון ב מחובר ל א דרך סינפסה B
מה יקרה לסינפסות אם נוירון ב יורה ומיד לאחריו נוירון א?
   - A תיחלש ו B יתחזק
2. באזור ה Paladus Ganglia מצאו כי כלל Anti Hebb עובד. מה יקרה לסינפסה בין 2 נוירונים שם אם הם יורים יחדיו?
   - יחלשו
3. לנוירון בקליפת המוח יש סף שלילי (ביאס חיובי) וסינפסות אינהיביטוריות והמסוגלות ליצר אך ורק זרמים שליליים (אינהיביטורים). הנוירון יורה במידה וסך הזרמים המתקבלים בדנדריטים גדול מהסף. מה יקרה לנוירון אם לא יתקבל שום קלט – ז.א. שום זרם לא יוזרק לדנדריטים.? מה יתקבל אם יתקבל זרם בעוצמה גוברת והולכת מהדנדריטים?
   - הנוירו ירה בקצב קבוע עד שיקבל זרם מהדנדיטים, ואז יקטין את קצב הירי עד להשתקתו
4. מה יקרה לנוירון אם יגיעו זרמים לדנטריטים אך סכום הזרמים קטן מהביאס?
   - ימשיך לירות אבל בקצב יותר נמוך
5. מה יקרה לנוירון אם סכום הזרמים בערך מוחלט גדול מהביאס?
   - הנוירון ישתתק
6. מה יקרה בנוירון ביאולוגי אקסיטטורי כאשר השני (post) יורה לפני הראשון (pre)? מה יקרה לנוירון מלאכותי כאשר מקטינים משקל חיובי עוד ועוד?
   - חוזק הסינפסה תיחלש עד שתגיע ל 0 (לא צעביר זרם) אולם לא תהפך לאינהיביטורית
   - בנוירון מלאכותי, חוזק הסינפסה תרד תחצה טת ה 0 ותהפך שלילית יוצר ויותר

1. איזו סוג רשת היא הכוללת 2 נוירונים הראשון מחובר לשני והשני מחובר לראשון. האם ניתן לתת קלט לרשת כזו?
   - RNN, כן: כל נוירון יכול לקבל עוד קלטים
2. אילו סוגי רשתות ניתן לתאר כ DAG?
   - FF
3. איזו תכונות ביאולוגית מבטאת בד"כ פונקציית האקטיבציה:
   - קצב ירי כפונקציה של סכום הזרמים הנכנסים בדנדריטים
   - הסתברות לירי כפונקציה של סכום הזרמים

Swish Self Gated Activation

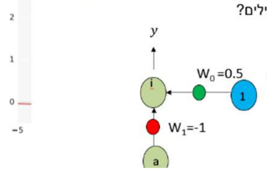
ניתן לבנות שערים לוגיים בעזרת רכיבי BTU (בדרך הנדסית)

- שער AND: y=a AND b

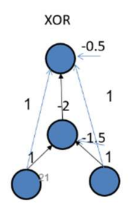


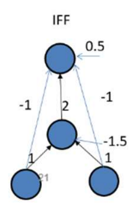
  – לציפור נוירון שמגיב לצבע סגול
  ונוירון שמגיב לצורה עגולה
  היא אוכלת רק דברים עגולים וסגולים
  (אוכמניה-כן, כדור גולף-לא)
  איזה מעגל עיצבי יממש נוירון שמגיב לדברים אכילים?

- שער NOT: y= ~a



– בנה רשת שמבצעת IFF
1. בעזרת 4 נוירונים נסתרים (משפט הבניה)
2. בעזרת 2 נוירונים נסתרים
3. בעזרת נוירון נסתר בודד (כולל מעקף מהקלט ישירות לפלט)



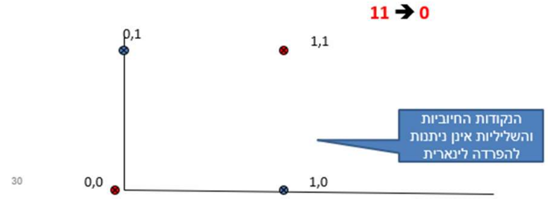


שימו לב, הנוירון הנסתר לא חייב לשנות את תפקידו. כדי להפוך את הפלט: משקולות הפלט מוכפלות ב -1.

Linear Inseparability of XOR Geometric View

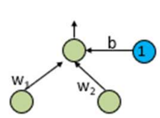
- במרחב קלט דו מימדי, האם הדוגמאות ניתנות להפרדה לינארית?

00 → 0
01 → 1
10 → 1
11 → 0



בקלט 2D, (יש 3 משקלים) כמה מיקרים בלתי תלויים לינארית צריך לכל הפחות על מנת שנוכל לבנות קבוצת אימון שאיננה ניתנת להפרדה?
תשובה: 4

הוכחה אלגברית:
לא ניתן לממש הפרדה לינארית ל XOR



00 → 0
01 → 1
10 → 1
11 → 0

$w_1x_1+w_2x_2+b=0$

– הוכחה בשלילה, נניח שניתן:
אזי קיימות משקולות שמפרידות את 4 הנקודות בטבלת האמת
נציב 4 הנקודות במשוואת ה BTU, נקבל 4 אילוצים סותרים:

$0w_1+0w_2+b \le 0$
$0w_1+1w_2+b > 0$
$1w_1+0w_2+b > 0$
$1w_1+1w_2+b \le 0$

$w_1+w_2 \le -b$; $w_1 > -b$; $w_2 > -b$; $b \le 0$

– ← $w_1+w_2 > -2b$ וגם $w_1+w_2 \le -2b$ ← סתירה!!

אלגוריתם: Perceptron Learning

עבור על כל דוגמאות הקלט שבקבוצת האימון שוב ושוב (עד שאין שגיאות)

1. חשב את y (החזוי) של הפרספטרון ע"פ וקטור הקלט x_j
2. חשב את השגיאה $(t_j - y)$
3. דחוף את וקטור המשקולות כך שהשדה המושרה יתקדם לכיוון הנכון ע"י הוספת $\Delta w_{ij} = \lambda(t_j - y_j)\, x_i$

מה ההיגיון?

1. אם Z_j קטן שווה מ 0, והמטרה $t_j=1$, השגיאה חיובית
   - אנחנו מתקנים את המשקל כך ש Z יגדל בכיוון הנכון
     אם הקלט חיובי מגדילים את המשקל ואם הקלט שלילי מקטינים.
2. אם Z_j גדול מ 0, והמטרה $t_j=0$, השגיאה שלילית
   - מתקנים את המשקל ומקטינים את Z בכיוון הנכון
     אם הקלט חיובי, מקטינים את המשקל ואם הקלט שלילי, מגדילים
3. האם התהליך יתכנס? האם לא יתכן שנתקרב למטרה בדוגמא אחת ונתרחק בדוגמאות אחרות?
   - רוזנבלט הוכיח ש**אם יש פיתרון, האלגוריתם ימצא אותו , אם אין פיתרון, האלגוריתם לא יתכנס אף פעם ואף תיתכן התבדרות של המשקולות**
   - במה תלויה הצלחת הלמידה?
   - אםם דוגמאות האימון ניתנות להפרדה לינארית!
     ז.א. אםם הקלט מכיל features "טובים" (הניתנים להפרדה לינארית)

האם כלל הלמידה של פרספטרון מזכיר לנו את הכלל ההביאני?

- למידה הבייאנית מתבטאת בשינוי משקלות הסינפסות:
  – הקשר מתחזק במידה i ו 2 נוירונים יורים ביחד

$$\Delta w_{ij} \sim y_j x_i$$

- למידה אנטי-הבייאנית:
  – הקשר נחלש אם הנוירונים יורים ביחד

$$\Delta w_{ij} \sim -y_j x_i$$

- האם כלל הפרספטרון הוא הביאני?:

$$\Delta w_{ij} \sim (t_j - y_j) x_i$$

- **המשקולת תשתנה רק אם החיזוי y לא זהה ל t**
  – את כאשר הקלט x_i חיובי ו $y=0$, המשקל יתחזק (הביאני)
  – כאשר הקלט שלילי ו $y=0$ אזי המשקל יחלש (אנטי הביאני)
  – כאשר הקלט חיובי ו $y=1$, המשקל יחלש (אנטי הביאני)
  – כאשר הקלט שלילי ו $y=1$ המשקל יתחזק (הביאני)

38

למידת פרספטרון: המשך תרגיל AND

- נתון פרסופטרון בעל 3 נוירוני קלט : x0,x1,x2 ונוירון פלט אחד. בצע למידה של הפונקציה AND בין 3 הקלטים כך שנוירון הפלט יהיה 1 אםם כל הקלטים הם 1.
- **המשקולות מתחילים ב (-1.2, -0.3, -0.4) וקבוע הלמידה הוא 0.9.**
- כלל העידכון:

$$\Delta w_{ij} = 0.9(t_j - y_j)\, x_i$$

t=רצוי y=חזוי

	x_0	x_1	x_2	w_0	w_1	w_2	output	desired output
	1	1	0	–1.2	–0.3	–0.4	0	0
	1	0	1	–1.2	–0.3	–0.4	0	0
	1	0	0	–1.2	–0.3	–0.4	0	0
הוספת (0.9,0.9,0.9)	1	1	1	–1.2	–0.3	–0.4	0	1
הפחתת (0.9,0.9,0)	1	1	0	–0.3	0.6	0.5	1	0
	1	0	1	–1.2	–0.3	0.5	0	0
	1	0	0	–1.2	–0.3	0.5	0	0
הוספת (0.9,0.9,0.9)	1	1	1	–1.2	–0.3	0.5	0	1
הפחתת (0.9,0.9,0)	1	1	0	–0.3	0.6	1.4	1	0

Gradient Descent Alogorithm for minimizing a loss function

- נתונה פונקצית עלות $loss_{D,h}(w_0, w_1, \ldots)$ שהגראדיינט שלה ניתן לחישוב
- נתונה קבוצות אימון D עם דוגמאות: $\langle x^i, t^i \rangle$
- רוצים למצוא $(w_0, w_1 \ldots)$ שימזער: $Min_{w0,w1\ldots}\{loss_{D,h}(w_0, w_1 \ldots)\}$
- העיקרון:
  – התחל מ $W = w_0, w_1 \ldots$ משקולות אקראיים
  – בצע Epocs שוב ושוב עד להתקימותו של תנאי העצירה.
    למשל: עד שהעלות מפסיקה לרדת, או שהגענו למקסימום Epocs

בכל Epoc:

- עבור D, חשב את הגראדיינט של פונקצית העלות בנקודה w
- שנה את $w_0, w_1 \ldots$ כך ש $loss(w_0, w_1 \ldots)$ יקטן:
  חישוב השינוי יהיה צעד כנגד כיוון השיפוע

$$\Delta w_i = -\lambda \frac{\partial loss_{D,h}(\mathrm{w}))}{\partial \mathrm{w_i}} = -\lambda \sum_{p \in D} \frac{\partial loss_D(y_p)}{\partial\, \mathrm{y_p}} \frac{\partial h_w(x_p)}{\partial \mathrm{w_i}} \qquad y = hw(x)$$

$$w_i = w_i + \Delta w_i$$

השיטה רלוונטית גם לפונקציות במימד גבוה וגם לפונקציות שאינן קמורות

Gradient Descent- Single var with no bias

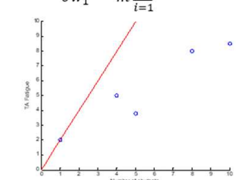
הנגזרת החלקית לפי w1:

$$\Delta w_1 = -\lambda \frac{\partial loss}{\partial w_1}$$

$$= -\lambda \frac{\partial loss}{\partial w_1} = \frac{\lambda}{m} \sum_{i=1}^{m} (t_p - y_p) x_p$$

נגדיר $\lambda = 0.01$

ונתחיל מ $w_1 = 2$



Gradient Descent- Single var with no bias

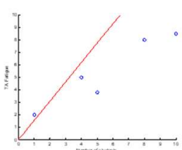
נבצע צעד GD:

$$\Delta w_1 = -\lambda \frac{\partial loss}{\partial w_1}$$

$$\frac{\partial loss}{\partial w_1} = \frac{1}{m} \sum_{p=1}^{m} (y_p - t_p) x_p$$

ערך התחלתי $w_1 = 2$

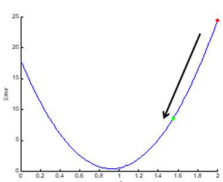
x_p	t_p	y_p	$(y_p - t_p)x_p$
10	8.5	2*10=20	(20-8.5)*10=125
5	3.8	2*5=10	(10-3.8)*5=31
1	2	2*1=2	(2-2)*1=0
8	8	2*8=16	(16-8)*8=64
4	5	2*4=8	(8-5)*4=12
$\frac{1}{m}\sum_{p=1}^{m}(y_p - t_p)x_p$:			46.4



צעד נוסף: עדכון המשקולות ע"פ ממוצע הגראדיינטים בנקודה החדשה

$$\Delta w_1 = -\lambda \frac{\partial loss}{\partial w_1} = -0.01 * 46.4 = -0.464$$

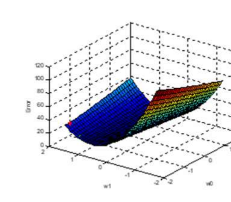
$$w_1^{new} = w_1 + \Delta w_1 = 2 - 0.464 \cong 1.54$$

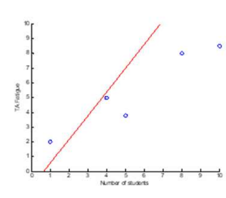


עדכון המשקולות וערך השגיאה

$$\Delta w_0 = -\lambda \frac{\partial loss}{\partial w_0} = -0.01 * 4.74 = -0.0474 \qquad \Delta w_1 = -\lambda \frac{\partial loss}{\partial w_1} = -0.01 * 38.8 = -0.388$$

$$w_0^{new} = w_0 + \Delta w_0 = -1 - 0.0474 \cong -1.05 \qquad w_1^{new} = w_1 + \Delta w_1 = 2 - 0.388 \cong 1.61$$





Stochastic GD: מדוע צריך לשפר את GD?

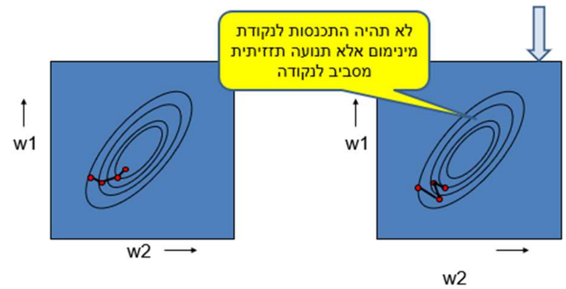
בשיטת ה GD מבצעים EPOCS שוב ושוב
בכל EPOC: מחשבים גראדיינט לכל דוגמא ובסוף ממוצע של הגראדיינטים כדי לבצע צעד שינוי משקולות קטן אחד

- GD הוא אלגוריתם off-line :
  משתמשים בכל ה training set (Epoc) כדי לבצע צעד אחד ב Batch
  אם קבוצת האימון גדולה, צריך למצע המוני גראדיינטים לפני שעושים צעד בודד
- **On-line Gradient Descent : וואריאציה של GD ל on-line - מבצעת צעד לאחר כל דוגמה**

Online (SGD) vs Batch (GD) walking in loss space

פונקצית השגיאה בנוירון לינארי היא קערה n מימדית- מינימום יחיד

- ב GD נתקדם במאונך לקווי הגובה של הקערה.
- ב SGD נתקדם בזיג-זג.



O/L SGD vs. GD יתרונות והסרונות

יתרונות: SGD

- מהיר יותר כאשר יש מיליונים של דוגמאות באימון.
  כל דוגמה מקדמת אותנו ולא צריך לסכם מיליונים לפני עדכון.
- כאשר יש כמה מינימומים מקומיים, התזזיתיות עוזרת לפעמים לברוח ממינימום מקומי
- ניתן להשתמש ב on-line learning
  (המשך הלמידה גם במצב production

חסרונות:

- מסלול תזזיתי אינו מביל ישירות למטרה (ב GD המסלול חמדני- הכי תלול),
- אין התכנסות (יש הסתובבות אינסופית סמוך למינימום)

גזירת ה- CEntropy

נוירון לוגיסטי: $z = wx = w_0 + \sum_i w_i x_i$ $\quad y=g(z)=\frac{1}{1+e^{-z}}$

$\frac{\partial z}{\partial w_i}=x_i$ $\quad \frac{\partial y}{\partial z}=\frac{\partial g(z)}{\partial z}= y(1-y)$

CEntropy על דוגמא אחת: $C(y,t)=-t\, ln(y)-(1-t)ln(1-y)$

$$\frac{\partial C}{\partial y}=(\frac{-t}{y}+\frac{1-t}{1-y})=\frac{-t(1-y)+y(1-t)}{y(1-y)}=\frac{-t+y}{y(1-y)}$$

$$\frac{\partial C}{\partial w_i}=\frac{\partial C}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i}=\frac{(y-t)}{y(1-y)}y(1-y)x_i=(y-t)x_i$$

On-line SGD: $\Delta w_i=-\lambda\frac{\partial C}{\partial w_i}=\lambda(t-y)x_i$

Mini-Batch SGD: $\Delta w_i=\lambda/m\sum_{p\in D}\lambda(t_p-y_p)x_{i\,i}$

עבור SGD/GD נרצה לחשב גראדיינט של ה loss כאשר $y=g(wx+b)$

- אם משתמשים ב MSE מקבלים:
  - פונקציה שאינה קמורה (יתכנו הרבה נקודות מינימום)
  - גראדיינטים שנעלמים
  - אין הצדקה ביאזינית ל MSE עבור קלסיפיקציה

כלל ה LMS של SGD מול כלל הלמידה של פרספטרון

ב GD
וב mini-batch
$$\Delta w_i=\frac{\lambda}{m}\sum_{p\in D}(t_p-y_p)x_{ip}$$

ב SGD
$$\Delta w_i=\lambda\,(t_p-y_p)\,x_{ip}$$

במה שונה מכלל הפרספטרון?

- Y לינארי (ולא פונקצית מדרגה)
- GD משתמש בממוצעים ואילו פרספטרון עובד דוגמא דוגמא
  Batch vs. on-line
- SGD מקרב ירידה עם גראדיינט שממזער פונקצית Loss.
  יש משמעות למינימום גם כשאין הפרדה לינארית מלאה
  למידת פרספטרון לא תעצור אם אין הפרדה לינארית

בפרספטרון, y היוא פונקצית המדרגה BTU
ברגרסיה y לינארי
On line של רגרסיה לינארית נהיה דומה לכלל הפרספטרון. ההבדל הוא: איך מחושב y
הירידה היא לא steepest אלא כל דוגמא, מושכת לכיוון שלה (יותר רועש, ואיטי)

תרגיל: איטרציה אחת של משקולות ב SGD m=1, עבור סיגמואיד בודד במרחב קלט 1D.

רוצים לסווג גידולים על פי גודלם.
נניח שהתחלנו ממשקולות: $w_1=-2,\ w_0=1$
קצב הלמידה $\lambda=1$
בהינתן דוגמה <x=2, t=1>

כיצד ישתנו המשקולות? $\quad \Delta w_i=-\lambda\frac{\partial C}{\partial w_i}=\lambda(t-y)x_i$

- חישוב קדימה: תחזית ההסתברות: $y=g(w_1x_1+w_0)=g(-2*2+1)=g(-3)=1/(1+e^3)\approx 0.05$
- חישוב השינויים במשקולות: $\Delta w_0=1(t-y)1=(1-0.05)=0.95$
  $\Delta w_1=1(t-y)x_1=1(1\text{-}0.05)2=1.9$
- חישוב המשקולות החדשים: $w_0=w_0+\Delta w_0=1+0.95=1.95$
  $w_1=w_1+\Delta w_1=-2+1.9=-0.1$

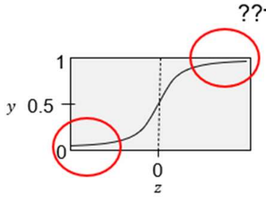
58

בעיית הגרדיאנטים הנעלמים

$$\frac{\partial MSE}{\partial w_i}=-\frac{1}{m}\sum_p y^p(1-y^p)(t^p-y^p)x_i^p$$

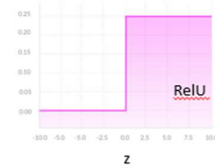
הגרדיאנט כופל את הפונקציה הלוגיסטית עצמה y^p וכן את הביטוי $1-y^p$.

מה יקרה כאשר y^p קרוב לאפס או אחד??



מה יקרה לפרוצדורת ה Gradient Descent?
מדוע ב CE לא נעלמים הגראדיינטים?

תרגיל: מהו כלל עידכון המשקולות עבור ReLU יחיד



$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i} = \frac{(y-t)}{y(1-y)}\frac{\partial y}{\partial z}x_i =$$

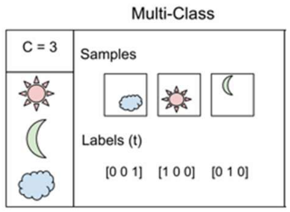
$$\frac{(y-t)}{y(1-y)}x_i$$ כאשר z>0

אחרת 0

$$\Delta w_i = \frac{\lambda}{m}\sum_p (t_p - y_p)y_p\,(1-y_p)\,x_{pi}$$

בבית: מהו כלל עידכון המשקולות עבור tanh ?

Multi-Class Cross Entropy



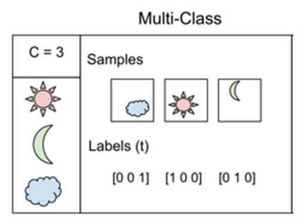
עבור דוגמא בודדה

$$MCC(y,t) = -\sum_{i}^{C=k}(t_i\log(y_i)+(1-t_i)\log(1-y_i))$$

עבור כל קבוצת האימון:

$$MCCE(y,t) = -\frac{1}{m}\sum_p\sum_i^k t_{pi}\log(y_{pi})+(1-t_{pi})\log(1-y_{pi})$$

Variation: Categorical Cross Entropy-



ב one-hot, רק השגיאה של הקטגוריה הנכונה נלקחת בחשבון.
בכל פעם שהחיזוי טועה, מגדילים את ההסתברות רק לנוירון שאמור להיות 1
מה החיסרון?

$$CCE(y,t) = -\sum_i^{C=k} t_i\log(y_i)$$

במיקרה הבינארי: יש זהות CE=CCE

$$-\sum_i^{C=2} t_i\log(y_i) = -t_1\log(y_1) - (1-t_1)\log(1-y_1)$$

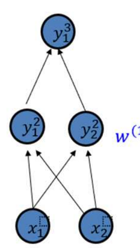
Multi class learning with Softmax units

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$ בשכבת הפלט כל נוירון מחשב:

$$\frac{\partial y_i}{\partial z_i} = y_i(1-y_i)$$ הנגזרת זהה לזו של הנוירון הלוגיסטי

- שיטה מקובלת לקלסיפקציה פרמטרית Multi Category:
  - מספר הפלטים זהה למספר הסיווגים (קטגוריות). מיצגים הסתברויות
  - סכום האקטיבציות בשכבת הפלט הוא 1
  - מתנהג בדומה לפונקצית MAX (אבל רך יותר)
  - כאשר ערך y של יחידת softmax אחת גדל , ערכי שאר היחידות קטנים

Feed forward – חישוב ערך השכבה הנסתרת



$$w^{(1)} = \begin{pmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} 0.1 & -0.2 & 0.3 \\ 0.2 & 0.1 & -0.3 \end{pmatrix}$$

$$z^{(2)} = \begin{pmatrix} 0.1 & -0.2 & 0.3 \\ 0.2 & 0.1 & -0.3 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.1 \\ 0.3 \end{pmatrix}$$

$$x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$z_1^{(2)} = 0.1 - 0.2 * 1 + 0.3 * 0 = -0.1$$

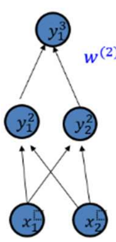
$$z_2^{(2)} = 0.2 + 0.1 * 1 - 0.3 * 0 = 0.3$$

$$y^{(2)} = g\begin{pmatrix} -0.1 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.47 \\ 0.57 \end{pmatrix}$$

$$y_1^2 = \frac{1}{1+e^{0.1}} = 0.47 \qquad y_2^2 = \frac{1}{1+e^{-0.3}} = 0.57$$

– Forward Computation
חישוב ערך הפלט ברשת

$$y_1^{(3)} = 0.56$$



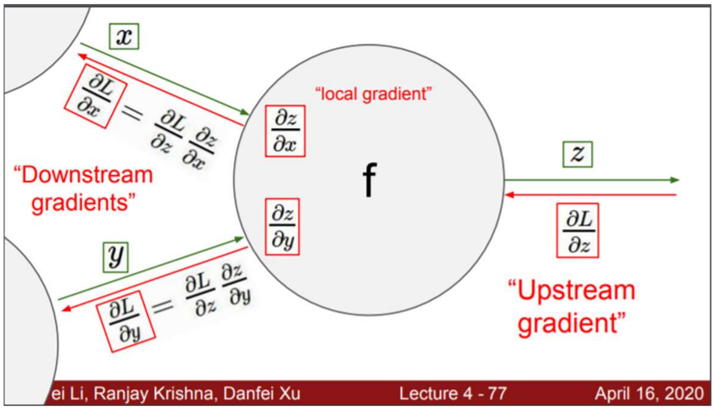
$$w^{(2)} = (0.1 \quad 0.2 \quad 0.1)$$

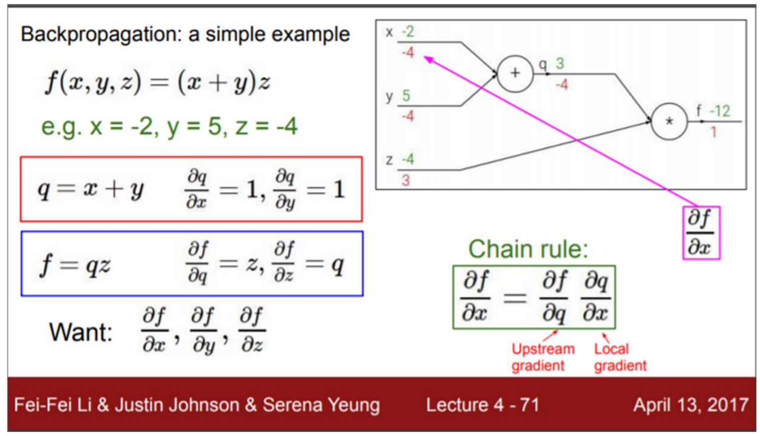
$$y^{(2)} = \begin{pmatrix} 0.47 \\ 0.57 \end{pmatrix}$$

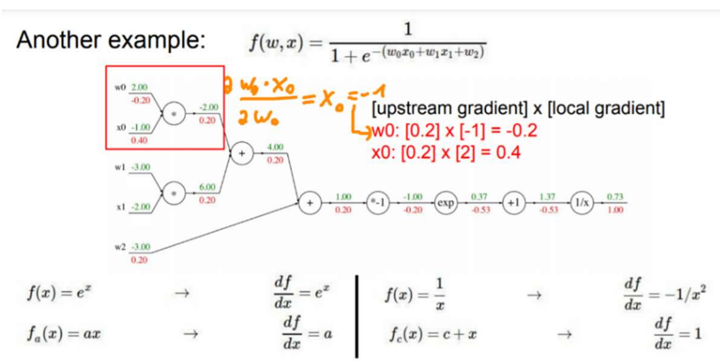
$$z^{(3)} = (0.1 \quad 0.2 \quad 0.1)\begin{pmatrix} 1 \\ 0.47 \\ 0.57 \end{pmatrix}$$

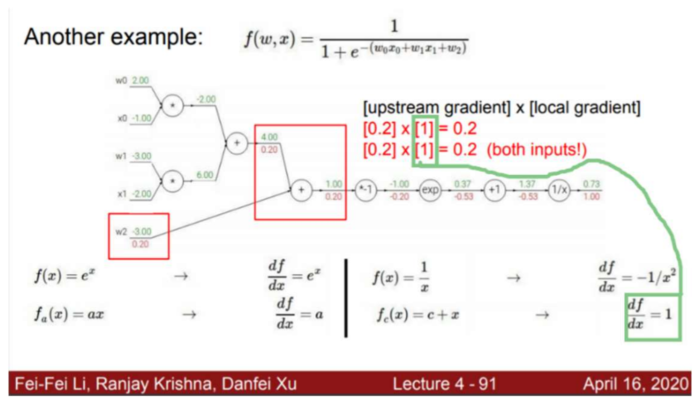
$$z_1^{(3)} = 0.1 + 0.2 * 0.47 + 0.1 * 0.57 = 0.25$$

$$y_1^3 = \frac{1}{1+e^{-z_1^3}} = \frac{1}{1+e^{-0.25}} = 0.56$$









AutoEncoders using BP

- Unsupervised learning
- מטרות:
  – Automatic feature discovery
  – Dimensionality Reduction - דחיסה ו יצוג אוביקטים דומים בעזרת וקטורים דחוסים המקימים "דימיון" וקטורי התואם "דימיון" אפליקטיבי
  – Semi-supervised learning

מדוע להשתמש בקלט OneHot בזבזני ולא ביצוג בינארי מקובל עם הרבה פחות מימדים

- במקום ליצג קלט אוביקט מסוים כ: (0,0,....0,1,0,0)
  Unit-לכל אוביקט: דרושים n units
  מדוע לא ליצג בצורה חסכונית יותר כ (0,0,1,1)?
  - דרושים רק log(n) units
- תשובה:
  – היצוג הבזבזני הוא בוודאות Linear separable ולכן היצוג הפנימי (המבוזר) בנקל יכול ליצור features רצויים לו
  – כאשר מקצים ביט לכל אוביקט לא מניחים שום ידע קודם. קידוד דחוס יותר חייב להניח דימיון בין חלק מהאוביקטים המיוצגים. ביצוג הבזבזני, כל האוביקטים (לא) דומים במידה שווה.
- האם ניתן ללמוד ייצוג דחוס המייצג שבו יצוגים דומים מייצגים אוביקטים דומים?

AutoEncoders for Unsupervised Learning

- סוג של למידה unsupervised, שמנסה לדחוס את המידע, למצוא יצוג פנימי ועל ידי כך לגלות features כלליים וטובים. אוביקטים עם מאפיינים דומים יקבלו ייצוג וקטורי דומה
  – למידה של פונקצית הזהות: הקלט שווה לפלט הרצוי.
  – בגלל צוואר בקבוק בשכבות הנסתרות, אין שינון של המידע אלא מתגלים features חשובים
  – אין מורה: למשל: ניתן לקחת תמונות (או הקלטות קול) ללא תוויות (או ללא תמלול)וללמוד להוציא מהם את המאפיינים החשובים שיאפשרו שיחזור של הקלט לאחר דחיסה

AutoEncoders for Internal representation

הרשת מוצאת יצוג פנימי מבוזר : מסבה Onehot encoding לקוד דחוס יותר

לאחר אימון, מקבלים יצוג ניסתר שונה לכל קלט. מה נקבל אם נעגל את ערכי היחידות הנסתרות ל 0/1?

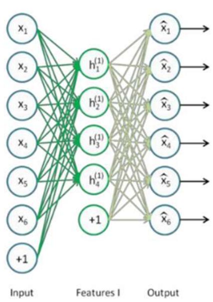
Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

28

Auto-encoders
האם ניתן לאמן רשת עמוקה ביעילות?

באמצע שנות האלפיים חיפשו טכניקות יעילות לאמן רשתות עמוקות (הימנעות ממינימום מקומי, האצת זמן האימון)



כאשר יש הרבה שכבות, משטח השגיאה מפותל מאוד,
כאשר יש רק שכבה אחת, פחות נתקעים במינימום מקומי.

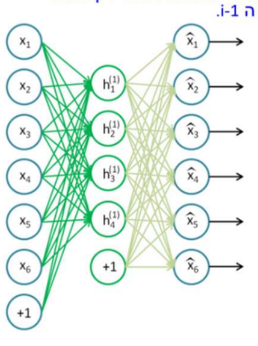
הרעיון: לאמן כל שכבה בנפרד, לקבל איתחול למשקולות, ואז לעשות Fine tuning

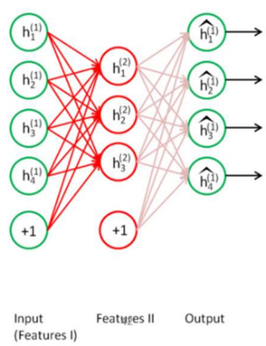
Bottle-neck layers in AutoEncoders

- דחיסה: השכבה האמצעית קטנה- צוואר הבקבוק מכביד על יכולת ההתאמה.
- Sparse Auto-encoder: מכבידים על הרשת בעזרת "עונשים" על האקטיבציה בשכבה הנסתרת
  (ולאו דווקא באמצעות דחיסה)
- Dropout הכבדה באמצעות הרג של נוירונים בשכבה הנסתרת

Stacked Auto-Encoders, Bengio (2007)
– After Deep Belief Networks (2006)

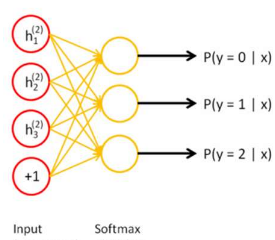
- שירשור חמדני של הרבה auto-encoders בטור.
- כל auto-encoder הוא בעל שיכבה נסתרת אחת. תוצאות ה encoding של רשת i הם הקלט (והפלט) לרשת i+1
- לאחר שאימנו auto-encoder i, נזרוק ממנו את ה decoder ונשאר רק אם ה encoder ואותו נשרשר לפלט של ה encoder ה i-1.

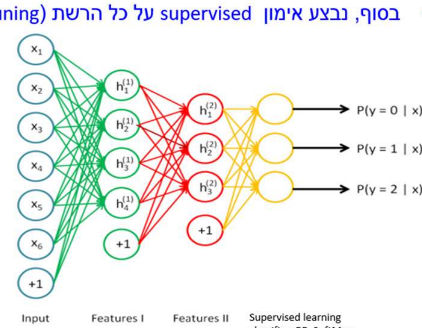




Stacked Auto-Encoders for Deep network initialization

- לאחר שיש לנו את שיכבת ה features האחרונה, נוסיף שכבת softMax (לקלסיפיקציה) ונאמן Supervised רק את המשקולות בשכבה זן.
- לאחר שאומנו כל השכבות בנפרד, מצרפים את ה encoders של של כל שכבה לרשת עמוקה, ועל השכבה העליונה, את שכבת ה softmax
- בסוף, נבצע אימון supervised על כל הרשת (fine-tuning לכל המשקולות)





43