עיבוד שפה טבעית ש7:
לקראת רשתות נוירונים

פתרון הבוחן

רענון רגרסיה לוגיסטית

- הפונקציה הלוגיסטית מעבירה מספר שרירותי למספר בין 0 ל-1 שאנחנו נקרא לו "הסתברות"
  - סיגמה מעל 0.5 = חיזוי 1, אחרת חיזוי 0

$$y = \sigma(z) = \frac{1}{1+e^{-z}}$$

- ה-z שבתוך הסיגמויד הוא מכפלת וקטור פיצ׳רים בוקטור משקלות נלמד (+ b נלמד)
- המטרה היא להתקרב עם כל חיזוי לתג האמיתי, ומינוס לוג ההפרש הוא ההפסד CE loss
- גזירת ההפסד לפי הפרמטרים נותנת, הודות לתכונות הסיגמויד, כלל עדכון מאוד פשוט:

$$\frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_j} = [\sigma(w\cdot x+b)-y]x_j$$

המשקולות (שייצגנו כ-θ)

הקלט (שייצגנו כ-f)

רענון ירידה סטוכאסטית בגרדיאנט (Stochastic Gradient Descent, SGD)

- מעבר על הדאטא דוגמא-דוגמא
- מצריך **קצב למידה** learning rate (יכול להיות בר-שינוי)
- מצריך **אתחול** כלשהו של הפרמטרים
- מצריך הגדרת **קריטריון התכנסות**

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J(\theta^{(t)})$$

פרמטרים
חדשים

קצב למידה

פרמטרים
קודמים

גרדיאנט ההפסד
(J יסומן אצלנו L)

אצוות זעירות (mini batches)

- עדכון הפרמטרים דוגמא אחרי דוגמא הוא לא יציב, ויכול להביא לתוצאות ביניים מאוד שונות
- הדוגמאות גם נבחרות אקראית, כך שתוצאות אימון יכולות להשתנות מאוד כתלות בסדר הבחירה
- הבעייה מתחדדת עוד יותר עם קצב למידה משתנה

- דרך התמודדות אחת: מעבר על סט האימון ב״נגלות״ של 10, 100, אפילו 1024 דוגמאות כל פעם
  - מחשבים את ההפרשים והגרדיאנטים של כל הדוגמאות לפי הפרמטרים הנוכחיים $\boldsymbol{\theta}^{(t)}$
  - מעדכנים את הפרמטרים לפי הגרדיאנטים שנאגרו בפעם אחת, בסוף האצווה
- בונוס יישומי: מאוד יעיל לחישוב מקבילי

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}).$$

רגולריזציה Regularization

- אצוות-זעירות מטפלות קצת ב״בריחת״ ערכי הפרמטרים, אבל לפעמים צריך לבלום אותם מפורשות
  - סכנה ל-overfitting

- **רגולריזציה** הוא אלמנט המתווסף לפונקציית ההפסד ו״מעניש״ אוספי פרמטרים (=מודלים) ש״הגזימו״: $L = L_{CE} + L_{REG}$
  - רגולריזציית L2: נתונה ע״י $L_{REG} = ||\theta||_2^2$, מענישה מודלים שהתרחקו יותר מדי מהראשית
  - רגולריזציית L1: נתונה ע״י $L_{REG} = ||\theta||_1$, מענישה מודלים שיותר מדי פרמטרים בהם אינם 0
  - מי מאלה נותנת כלל עדכון יותר נוח?
- לרוב נמשקל את הרכיבים: $L = L_{CE} + \lambda L_{REG}$

היפר-פרמטרים

- נתקלנו כבר בכמה גדלים שיכולים להשפיע על ביצועי המודל שלנו אבל לא ניתנים ללמידה ישירות בתהליך האימון של SGD:
  - קצב הלמידה **η**
  - משקל הרגולריזציה **λ**
  - גודל מיני-אצווה **m**
- אלה נקראים **היפר-פרמטרים** ונלמד אותם בנפרד מהאימון הרגיל
- תהליך הלמידה שלהם נקרא כיוונון (**tuning**) ולכבודו נקים עוד פיצול בדאטא שלנו, נוסף על train ו-test. שמות מקובלים הם **dev**(elopment) ו-**val**(idation)
  - יחס סדר-גודל מקובל עבור החלקים: 80% אימון, 10% ולידציה, 10% מבחן (ייתכנו אחרים)
  - בשקפי העזר על לימוד מכונה מוסבר מהו cross-validation, מומלץ לעיין

רגרסיה מרובת-סוגים (Multinomial regression)

- עד עכשיו התמודדנו עם שני קלאסים, הרבה פעמים נצטרך יותר
- בנינו פונקציית הסתברות ע״י שימוש במספר אחד -- וגם במקרה החדש יהיו לנו דרגות חופש כמספר הקלאסים **פחות אחד** (כי אנחנו עדיין רוצים פונקציית הסתברות)
- הדרך שלנו ״לחלק את העומס״ - נחשב ציון לכל קלאס, ובסוף ננרמל בסכום כל הציונים
- קיבלנו את פונקציית הסופטמקס (softmax)

$$\mathrm{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)} \quad 1 \leq i \leq k$$

שימו לב ל**תחום** ול**טווח** שהוא כל הכניסות בווקטור ⇐ מקבלת וקטור, מחזירה וקטור

למה קוראים לזה softmax?

- וקטור עם גורם מקסימלי ״מובהק״ - [9,4,0,0,2,3,1]
  - מחזיר כמעט הכל על המקסימום - [0.9894,0.0067,0.0001,0.0001,0.0009,0.0025,0.0003]
- וקטור עם כמה גורמים דומים למקסימום - [9,8,8.5,0,2,3,1]
  - עדיין מבהיר מי המקסימום, אבל עם פחות בטחון - [0.5055,0.1860,0.3066,0.0001,0.0005,0.0013,0.0002]
- זכרו, התחום הוא כל הממשיים!
  - [2, -9, -100, 0, 0, 1, 1]
  - [0.4984, $8.3\text{x}10^{-6}$, $2.8\text{x}10^{-45}$, 0.06745, 0.6745, 0.1834, 0.1834]

איך מקבלים ציונים שונים לכל קלאס?

- במקרה של שני קלאסים נדרשנו רק לציון אחד, שהגדירה לנו מכפלת **θ** ב-**f**.
- כאן צריך כמה ציונים, אבל **f** זהה עבור כל דוגמה.
- פתרון: נלמד **θ** נפרדת לכל קלאס:

$$x_5 = \begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \qquad \mathrm{w}_5 = 3.0$$

Feature	Definition	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	-5.3

בכמה סיבכנו את המודל?

מולטיקלאס - הפסד

- במקרה הבינארי, היה לנו סכום "ממושקל" של שני מקרים
  - "ממושקל" במרכאות כי אנחנו עובדים עם ברנולי: רק אחד מהם שווה 1, השאר 0.

$$L_{\mathrm{CE}}(\hat{y},y) = -\log p(y|x) \;=\; -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

מולטיקלאס - הפסד

- במקרה הבינארי, היה לנו סכום "ממושקל" של שני מקרים
  - "ממושקל" במרכאות כי אנחנו עובדים עם ברנולי: רק אחד מהם שווה 1, השאר 0.

$$L_{\mathrm{CE}}(\hat{y},y) = -\log p(y|x) \;=\; -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

- כשנכליל למקרים מרובים נגדיר את מרחב הקלאסים כוקטור אינדיקטור, ונקבל:

$$L_{\mathrm{CE}}(\hat{y},y) \;=\; -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

  - שוב, רק אחד מה-y_k הוא 1, ולכן, **עבור k הנכון**,

$$= \; -\log \hat{y}_k \;=\; -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

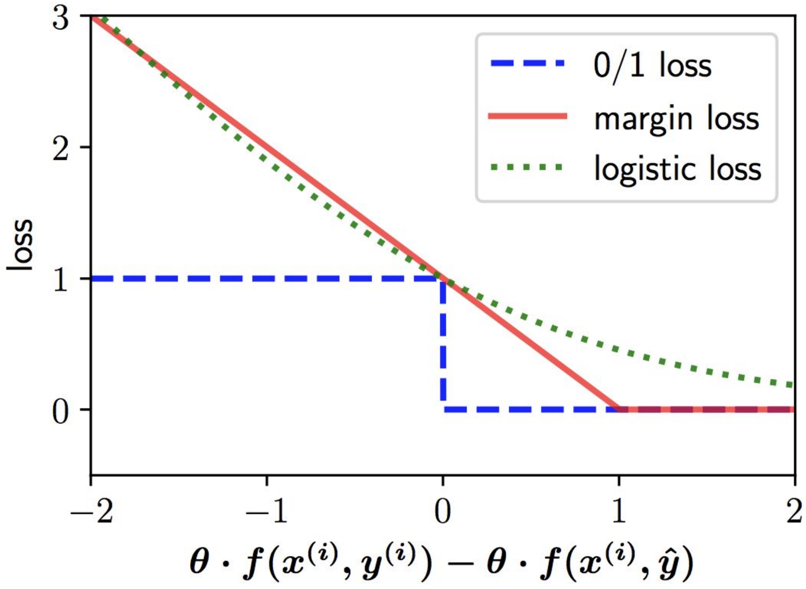
מולטיקלאס - כלל העדכון

- דומה למקרה הבינארי (או להכללה שלו לפי ערכי y)

- שימו לב ש**כל** הפרמטרים מתעדכנים
  - של הקלאסים השגויים (כמה? למה?)
  - של הקלאס הנכון (כמה? למה?)

$$
\begin{aligned}
\frac{\partial L_{\mathrm{CE}}}{\partial w_{k,i}} &= -(\mathbb{1}\{y=k\} - p(y=k|x))x_i \\
&= -\left(\mathbb{1}\{y=k\} - \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}\right)x_i
\end{aligned}
$$

פרספטרון Perceptron



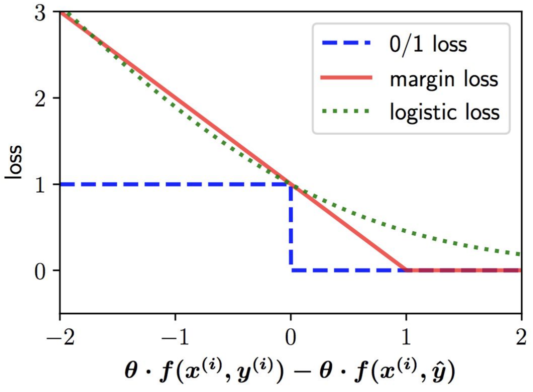
- מודל סיווג לינארי שלא עובד עם הסתברויות. הוא מנסה ״לצייר קו מבחין״ בין דוגמאות חיוביות לשליליות.
- הפרמטרים הם אותם פרמטרים כמו ברגרסיה לוגיסטית, והחישוב זהה מלבד הסיגמויד: חוזה "1" אם ורק אם **fθ**+b>0
- הפסד: ״הפסד 0-1״ (zero-one loss)

פרספטרון - כלל העדכון

- **אם צדקנו - אין עדכון**
- אם טעינו - נעדכן לפי "מה שהיה אמור לקרות", כלומר לפי הקלאס הנכון
- במקרה המולטי-קלאס, זה יוצא כך:

procedure PERCEPTRON($\boldsymbol{x}^{(1:N)}, y^{(1:N)}$)
 $t \leftarrow 0$
 $\boldsymbol{\theta}^{(0)} \leftarrow \mathbf{0}$
 repeat
  $t \leftarrow t+1$
  Select an instance i
  $\hat{y} \leftarrow \operatorname{argmax}_y \boldsymbol{\theta}^{(t-1)} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y)$
  if $\hat{y} \neq y^{(i)}$ **then**
   $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) - \boldsymbol{f}(\boldsymbol{x}^{(i)}, \hat{y})$
  else
   $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)}$
 until tired
 return $\boldsymbol{\theta}^{(t)}$

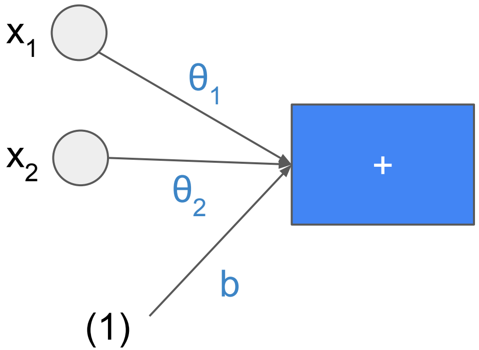
פרספטרון + הפסד מְרווח = SVM

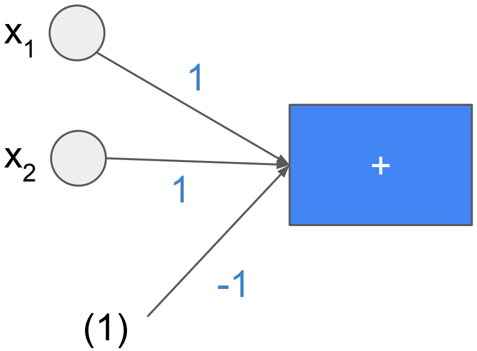


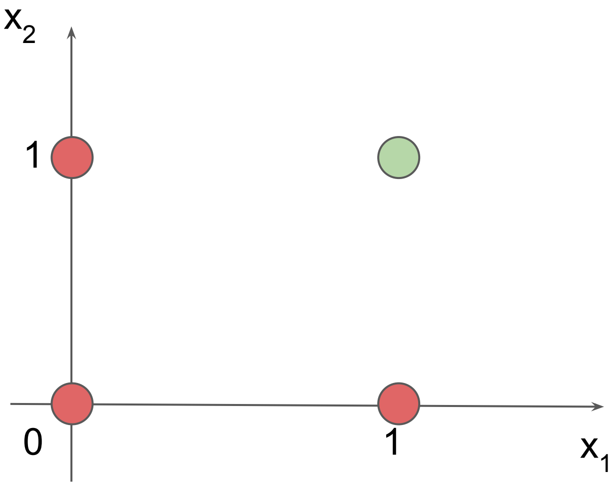
- מה לגבי margin loss שראינו קודם? (עוד שם - hinge loss)
- מטרה - למצוא גבול לינארי ״יעיל ככל האפשר״ = רחוק ככל הניתן מאיזשהי דוגמה בסט האימון
  - הדוגמאות הכי קרובות מכל קלאס נקראות וקטורי תמיכה, מכאן Support Vector Machine
- פרספטרון רגיל לא יכול למצוא כזה כי אין עדכונים כשצודקים
- מצד שני, זה יכול להועיל נגד overfitting
- (לא נרחיב על הגזירות ואופן האימון, אפשר לעיין באייזנסטיין 2.4.1-2.4.2)

רשתות נוירונים (פשוטות)

- יחידת הבסיס ("נוירון") - פרספטרון (ציון לינארי, חיזוי לפי > 0)
- נדמיין פרספטרון שצריך לחשב AND

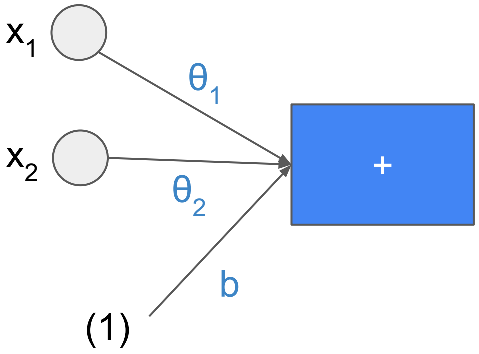


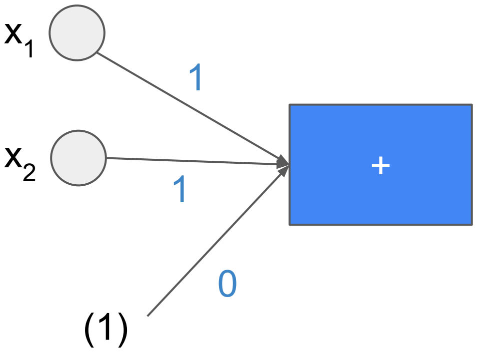


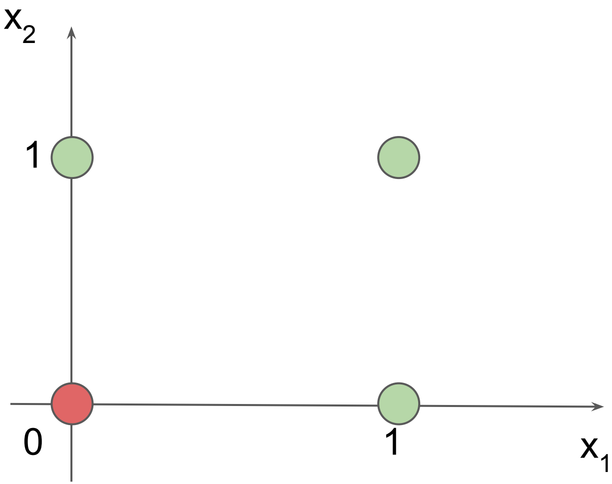


רשתות נוירונים (פשוטות)

- יחידת הבסיס ("נוירון") - פרספטרון
- נדמיין פרספטרון שצריך לחשב AND
- וכעת פרספטרון שצריך לחשב OR

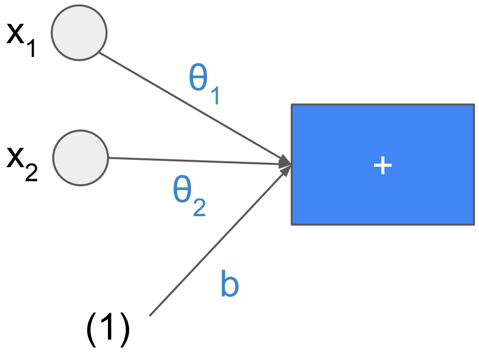


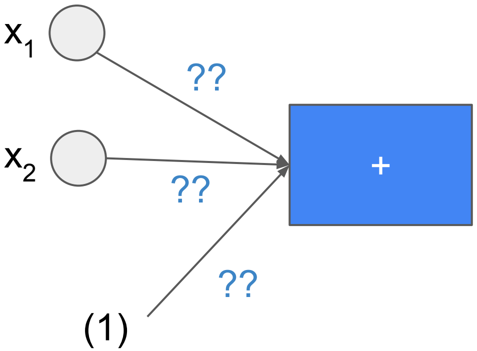


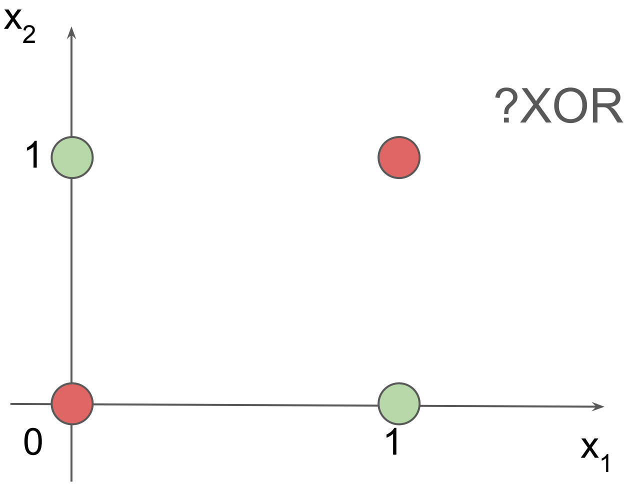


רשתות נוירונים (פשוטות)

- יחידת הבסיס ("נוירון") - פרספטרון
- נדמיין פרספטרון שצריך לחשב AND
- וכעת פרספטרון שצריך לחשב OR
- מה לגבי XOR?

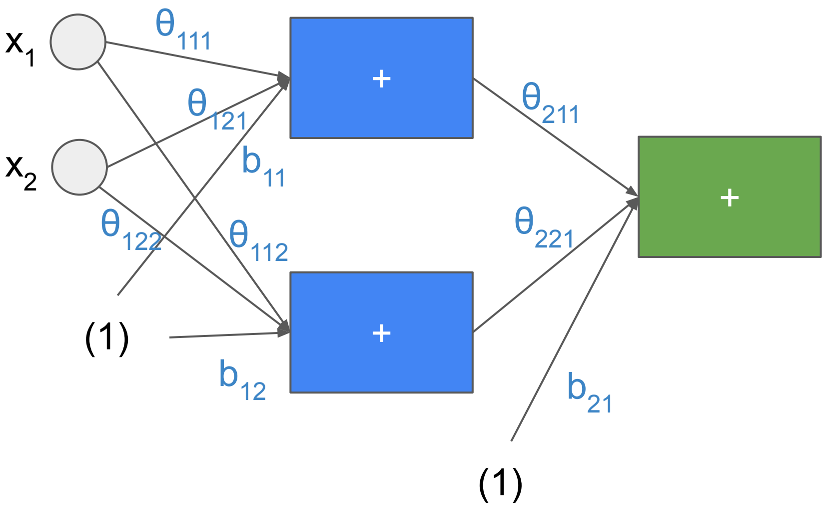






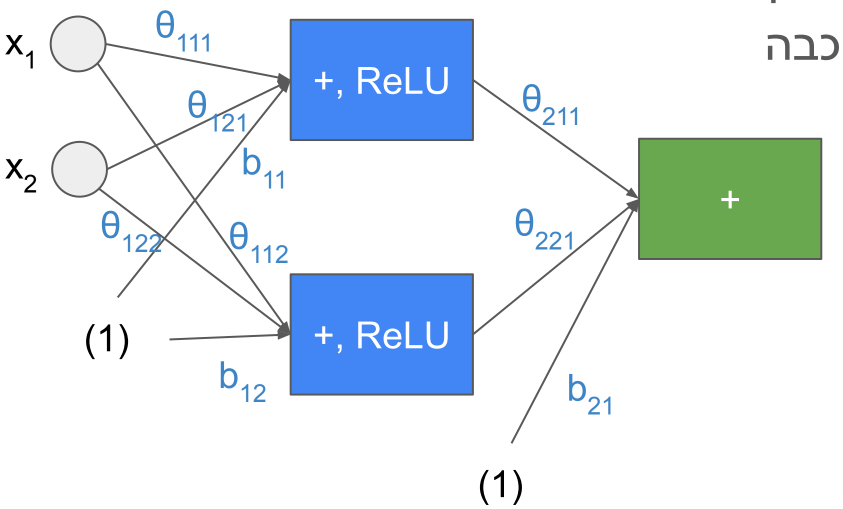
רשתות נוירונים (פשוטות)

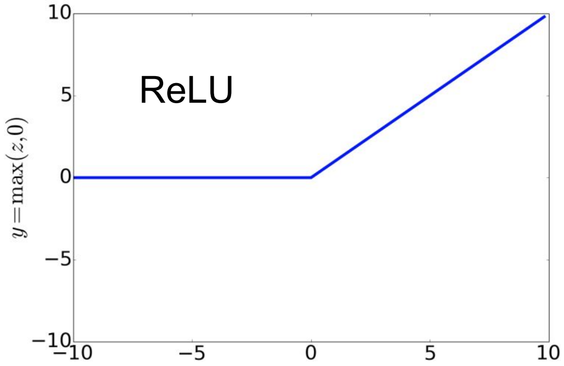
- נרכיב שלושה פרספטרונים אחד על-גבי השני



רשתות נוירונים (פשוטות)

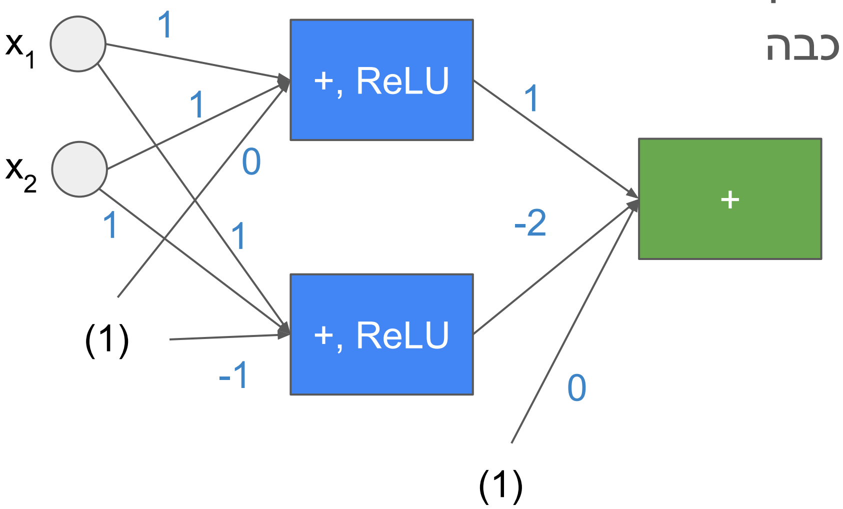
- נרכיב שלושה פרספטרונים אחד על-גבי השני
- סתם חיבור לא יעזור (למה?) ולכן נוסיף **אקטיבציה** בשכבה האמצעית (השכבה הנחבאת, hidden layer)

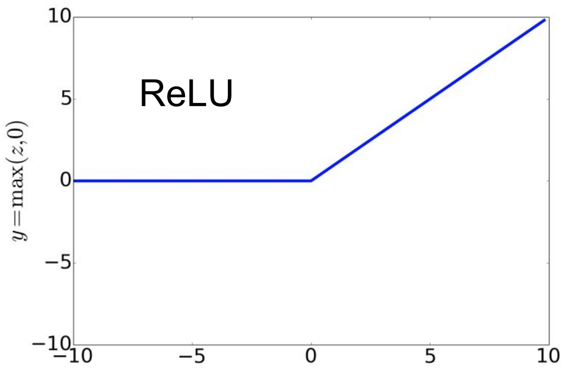




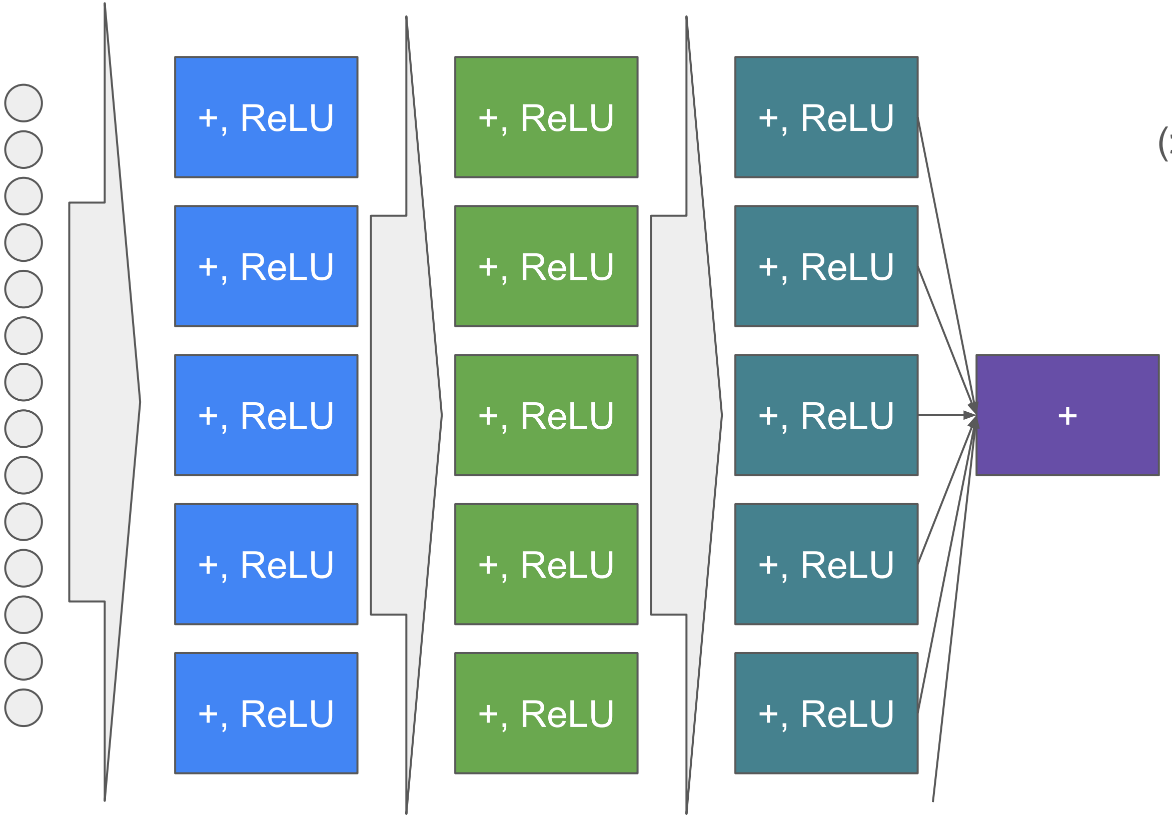
רשתות נוירונים (פשוטות)

- נרכיב שלושה פרספטרונים אחד על-גבי השני
- סתם חיבור לא יעזור (למה?) ולכן נוסיף **אקטיבציה** בשכבה האמצעית (השכבה הנחבאת, hidden layer)





רשתות נוירונים בהיזון קדמי (Feedforward NNs, FFN, MLP)

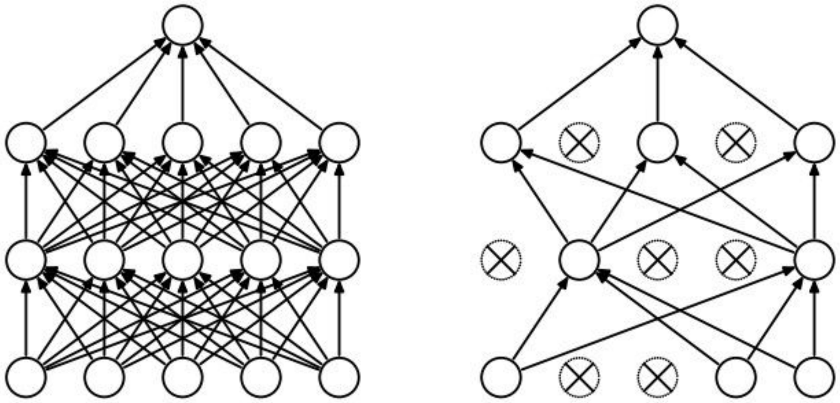


(1)

- אותו דבר רק בהמון
  - המון שכבות (= רשתות **עמוקות**)
  - המון פיצ׳רים
  - המון נוירונים חבויים בכל שכבה
- איך מאמנים אותן?
  - פעפוע אחורה - backpropagation
  - איך מאתחלים את המשקלות?
- איזו אקטיבציה בוחרים?
  - ReLU
  - סיגמויד
  - טנגנס היפרבולי tanh
- אילו פיצ׳רים מכניסים?

רגולריזציה ברשתות נוירונים - Dropout

- הגבלה ישירה על גודל כלל הפרמטרים (כמו ב-L_{REG} שראינו קודם) מכבידה על הלמידה ולא ברור אם היא יעילה
- במקום זה, מכניסים אלמנט הסתברותי לתהליך הלמידה: בכל שלב נאפס חלקים אקראיים בכל שכבה
- הרשת "לומדת להסתדר" בלי כל אחד מהנוירונים
- בזמן ריצה (inference), לא מחילים dropout



(a) Standard Neural Net (b) After applying dropout.

מקור התרשים:
https://deepnotes.io/dropout

רשתות נוירונים בהיזן קדמי - הדגמה

- https://bit.ly/3Dnm52U
- השימוש העיקרי ל-FFNs בשפה טבעית - **סיווג**.