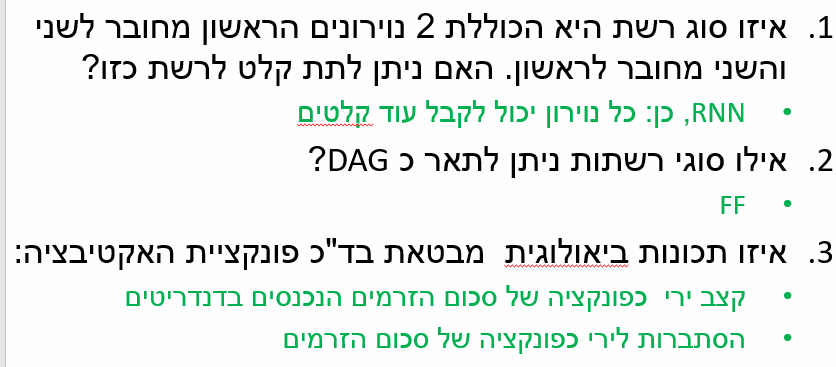
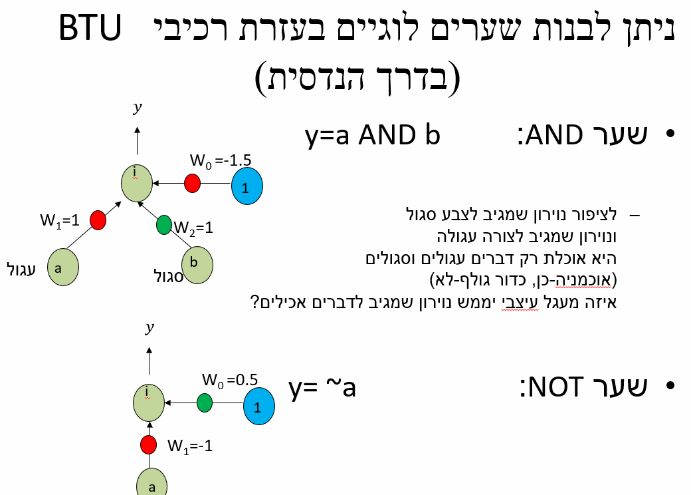
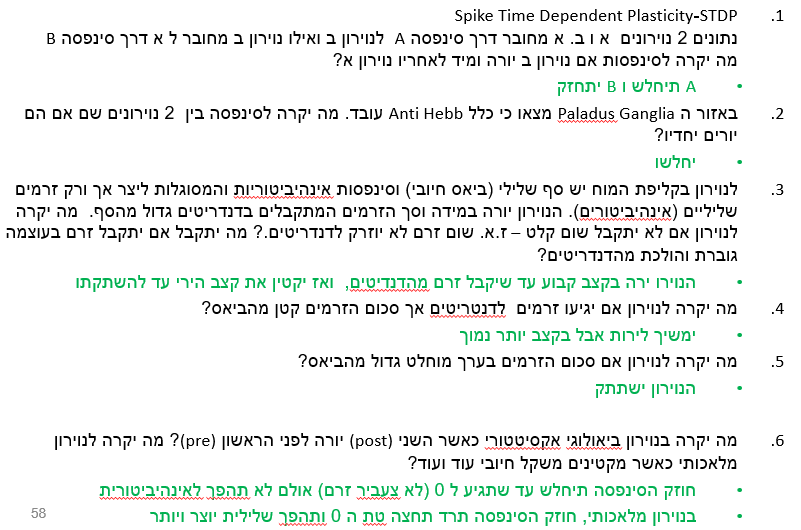
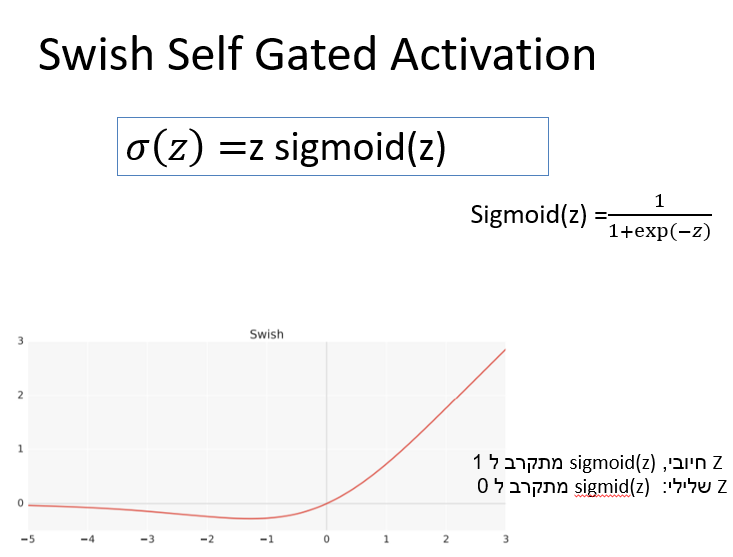
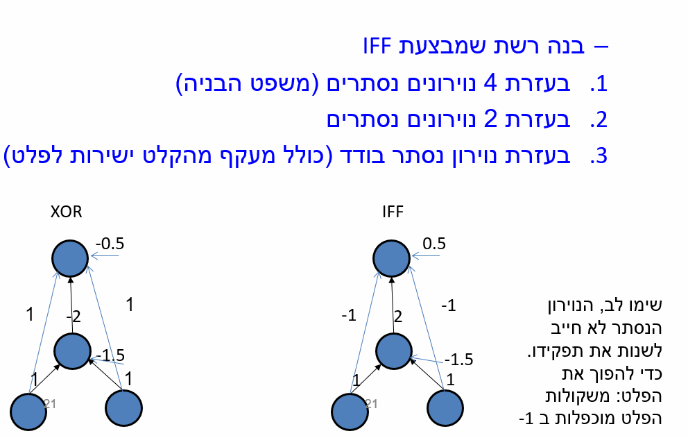
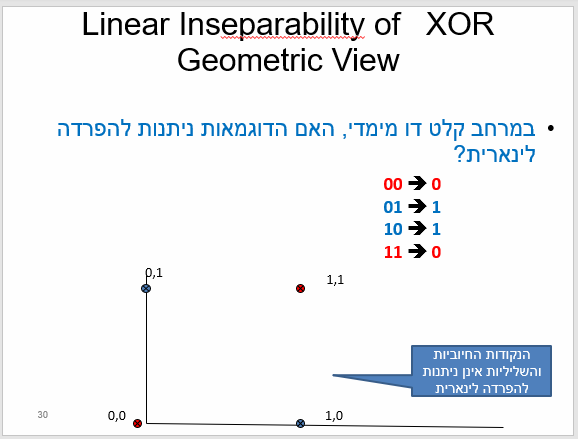
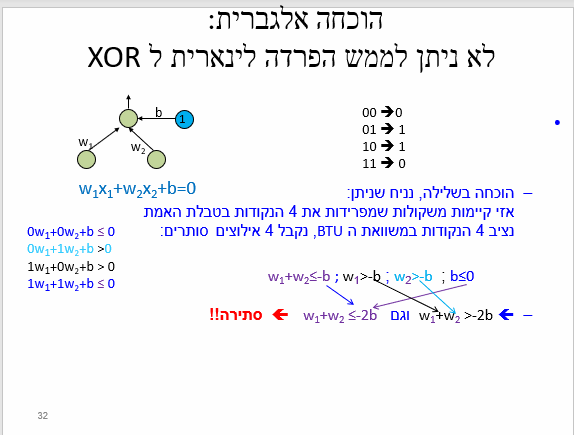
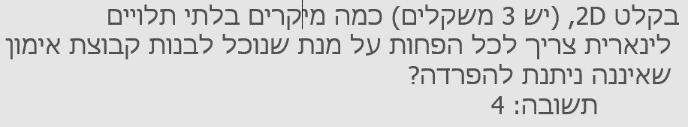
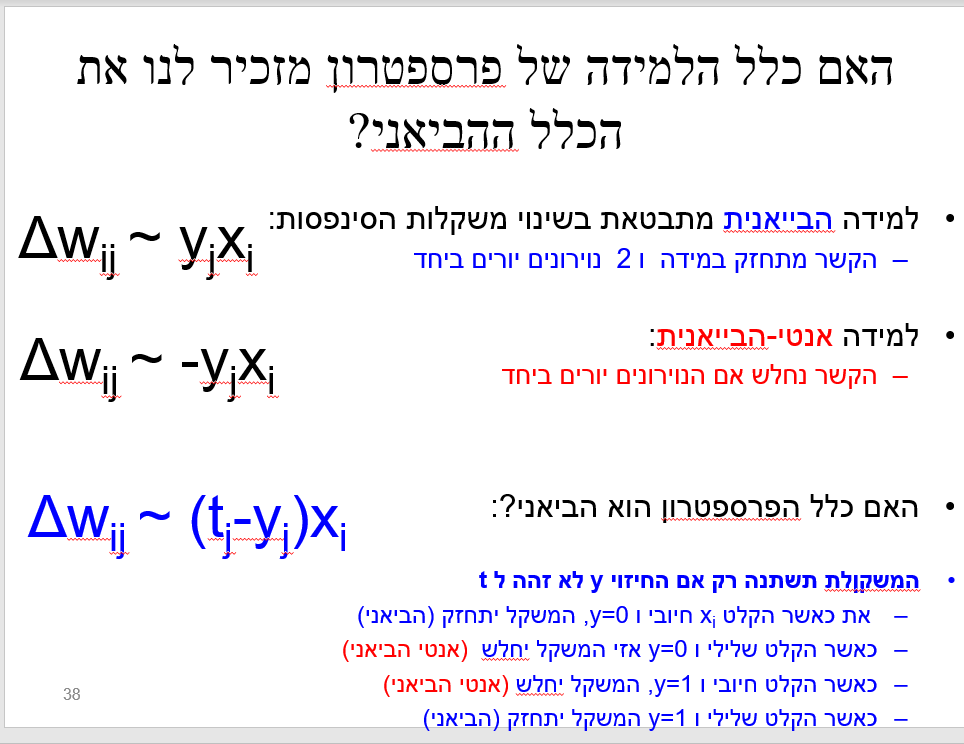
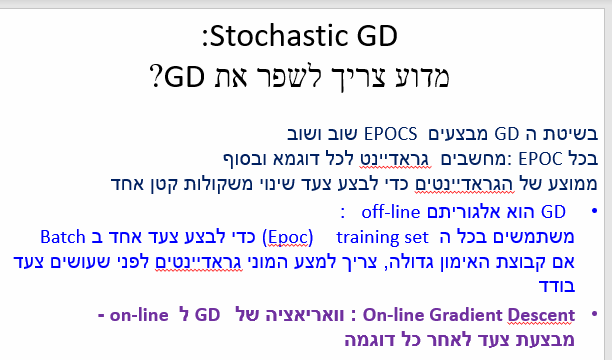
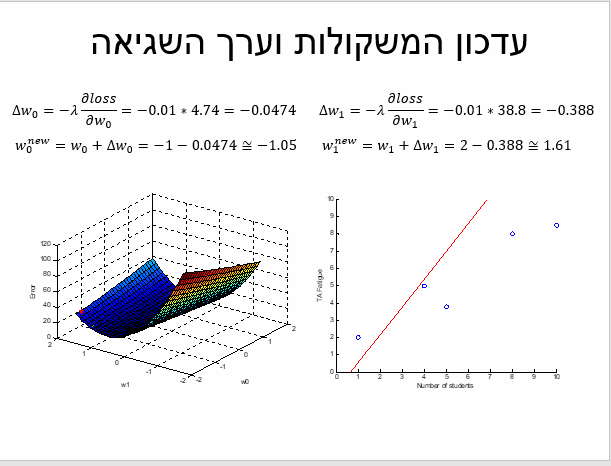
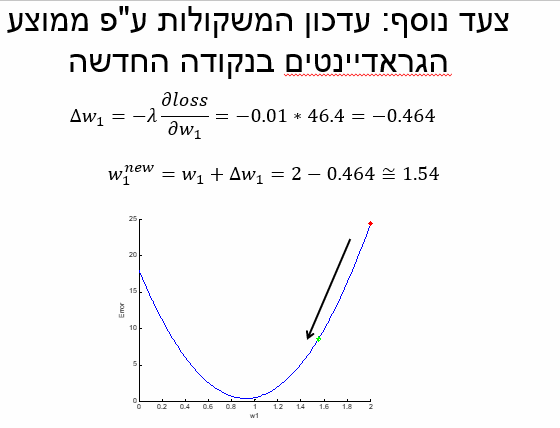
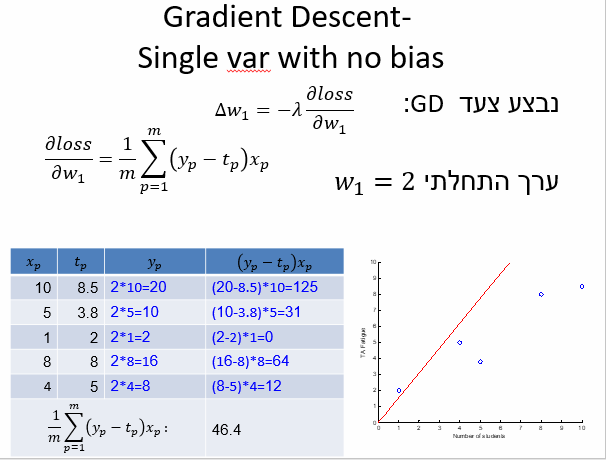
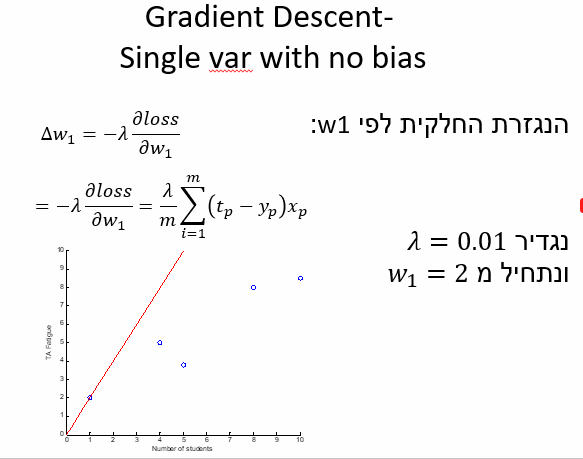
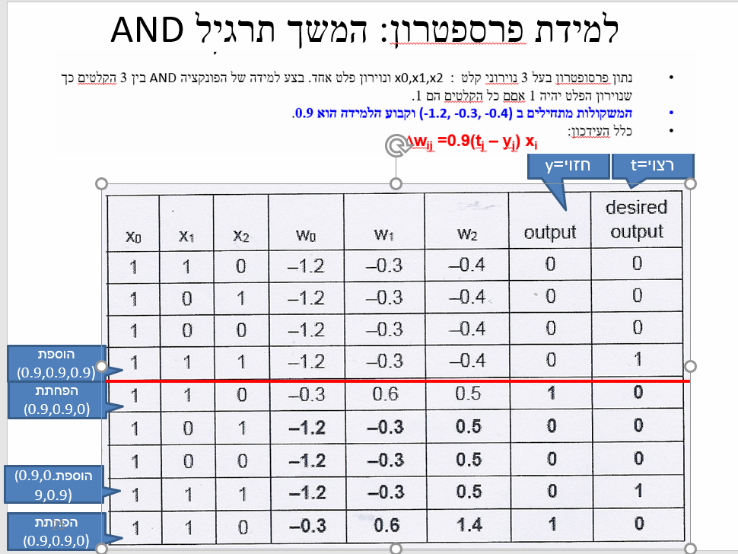
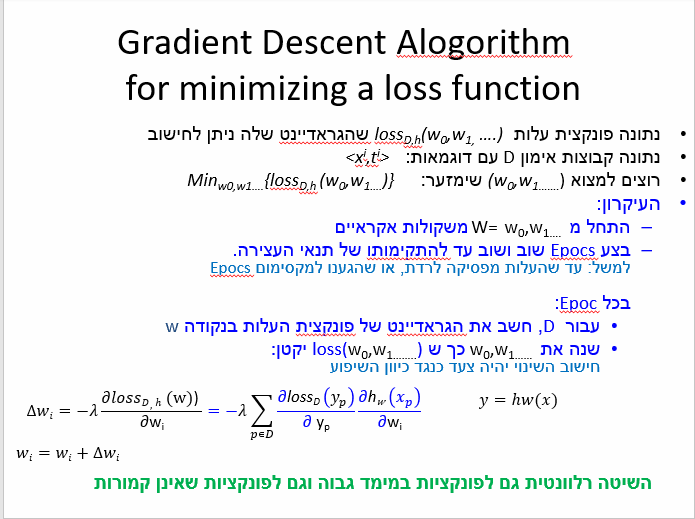
**** **** 

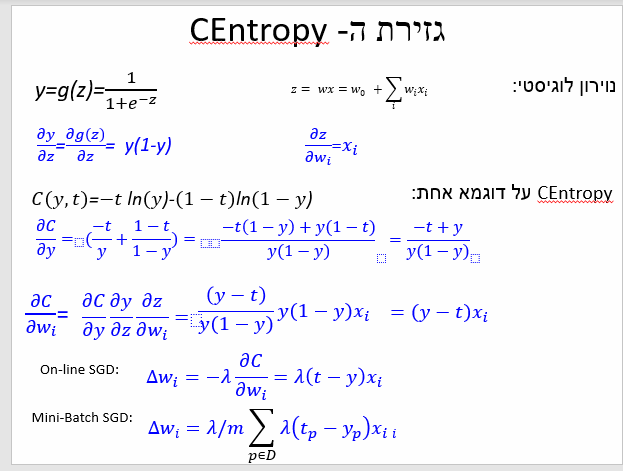
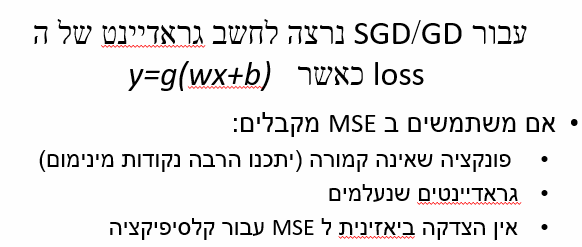
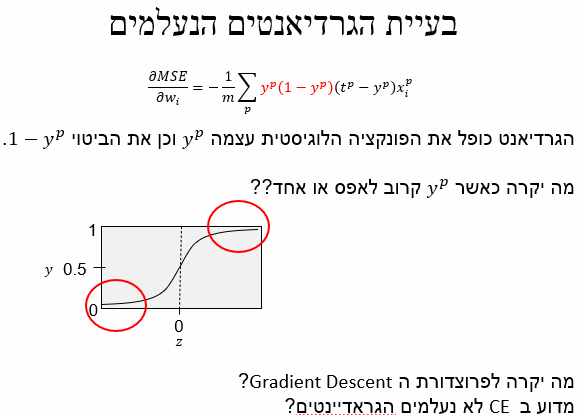
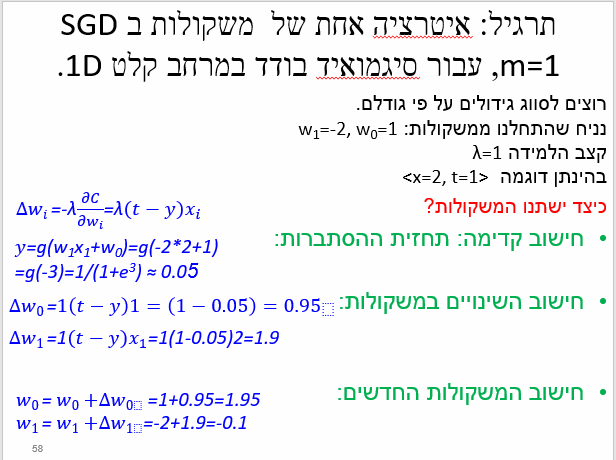
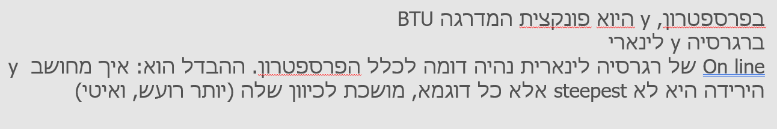
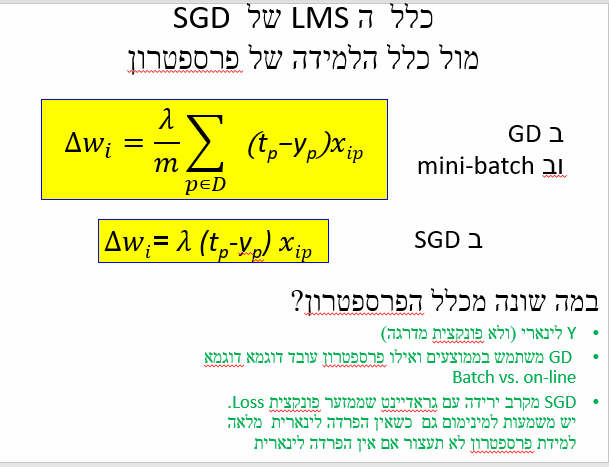
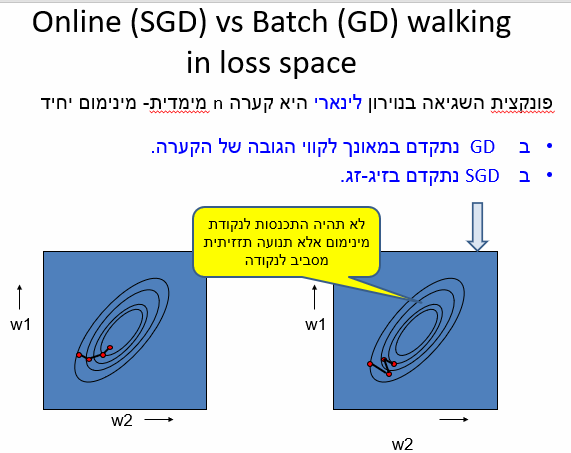
****

****

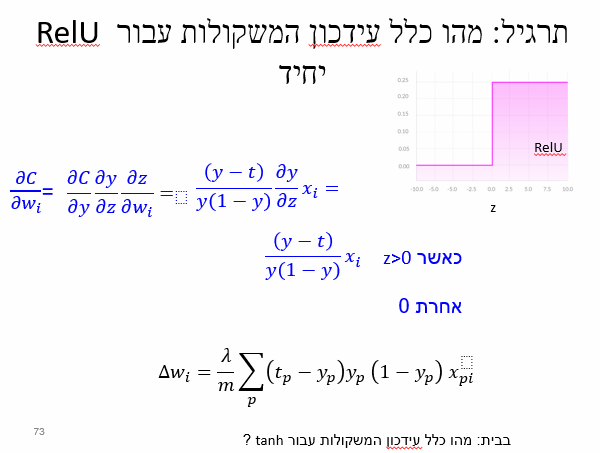
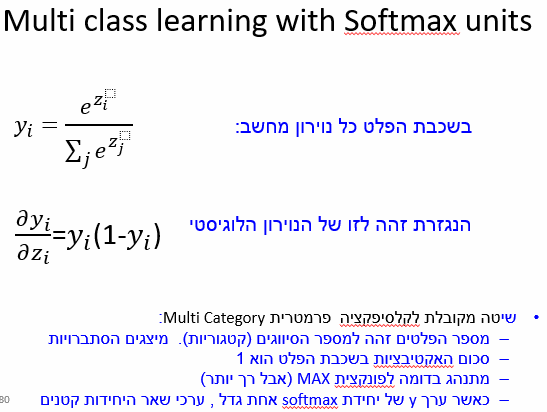
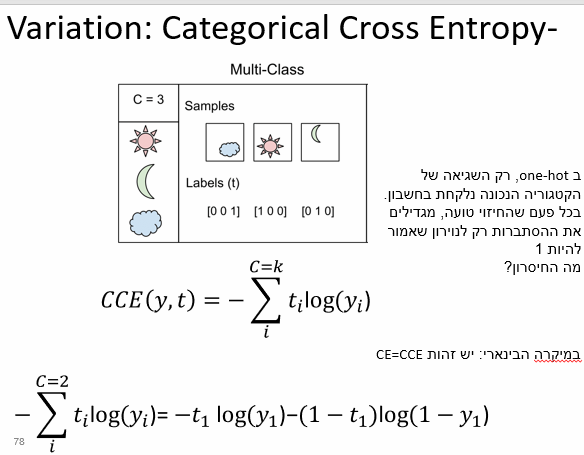
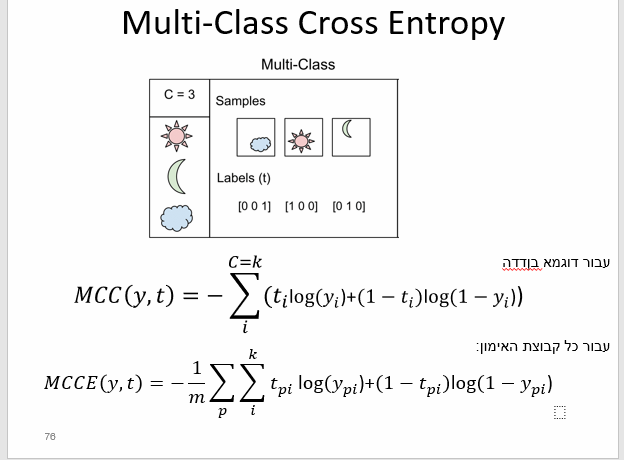
****

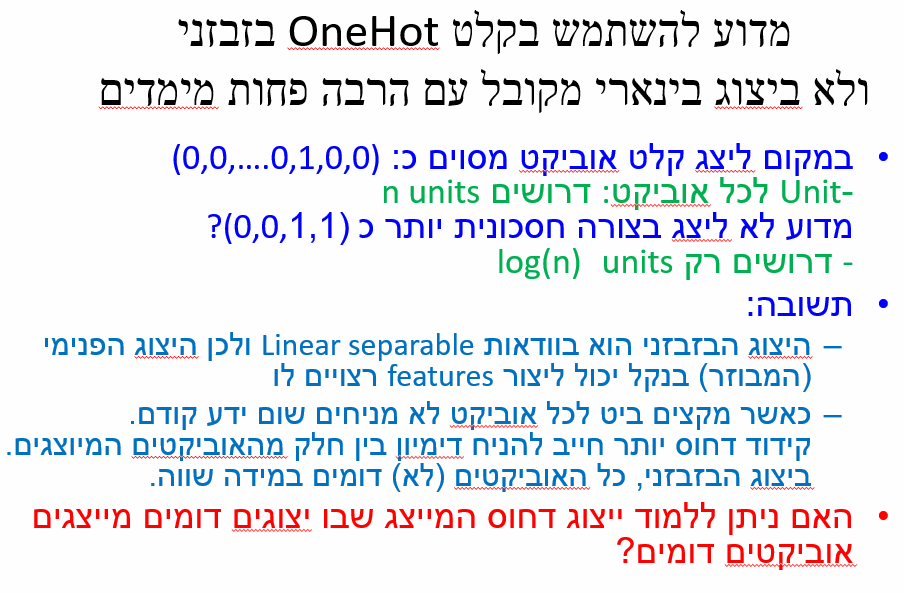
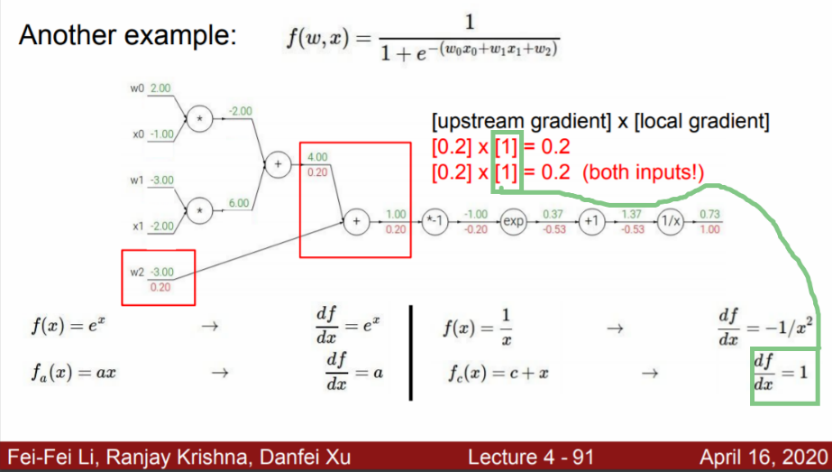
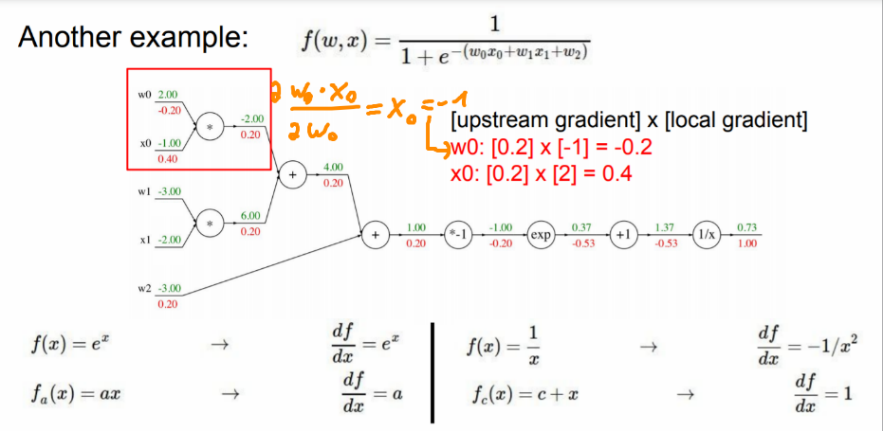
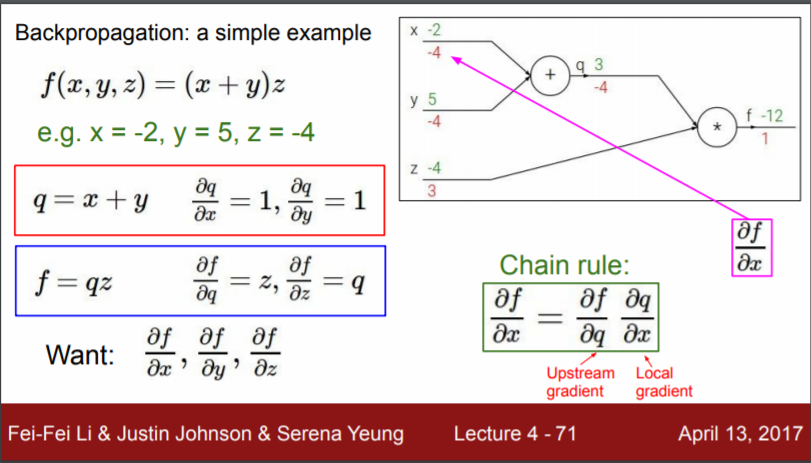
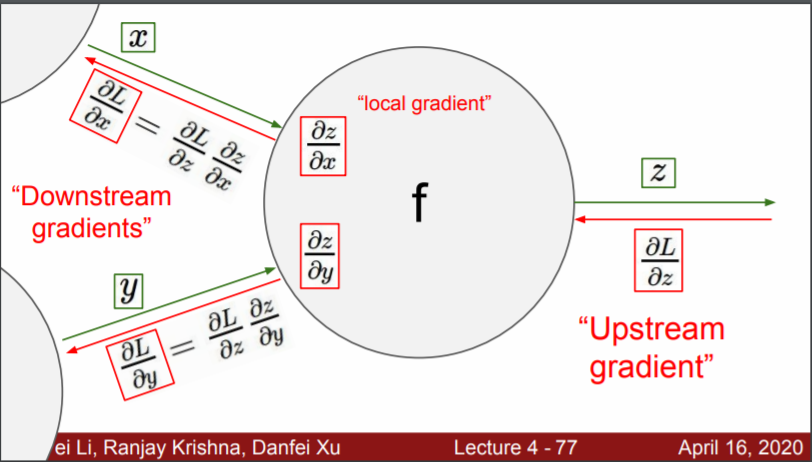
****

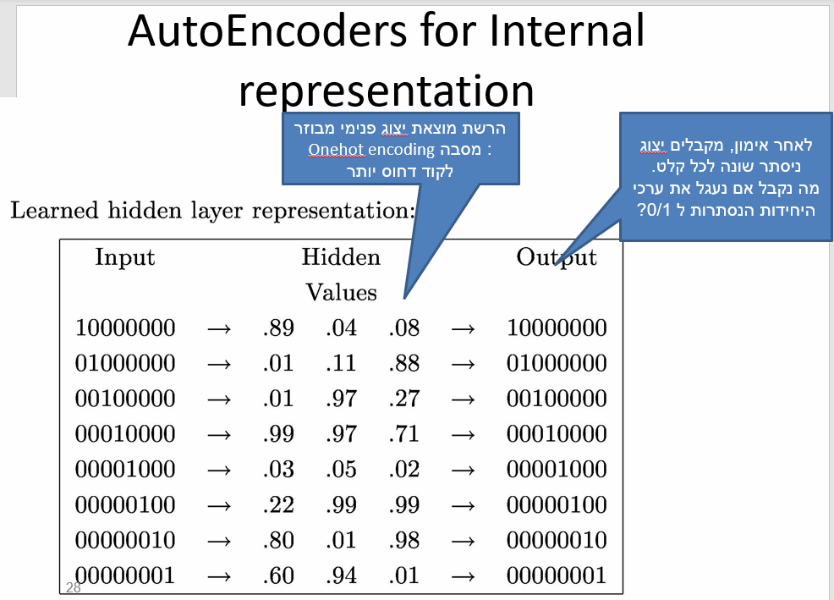
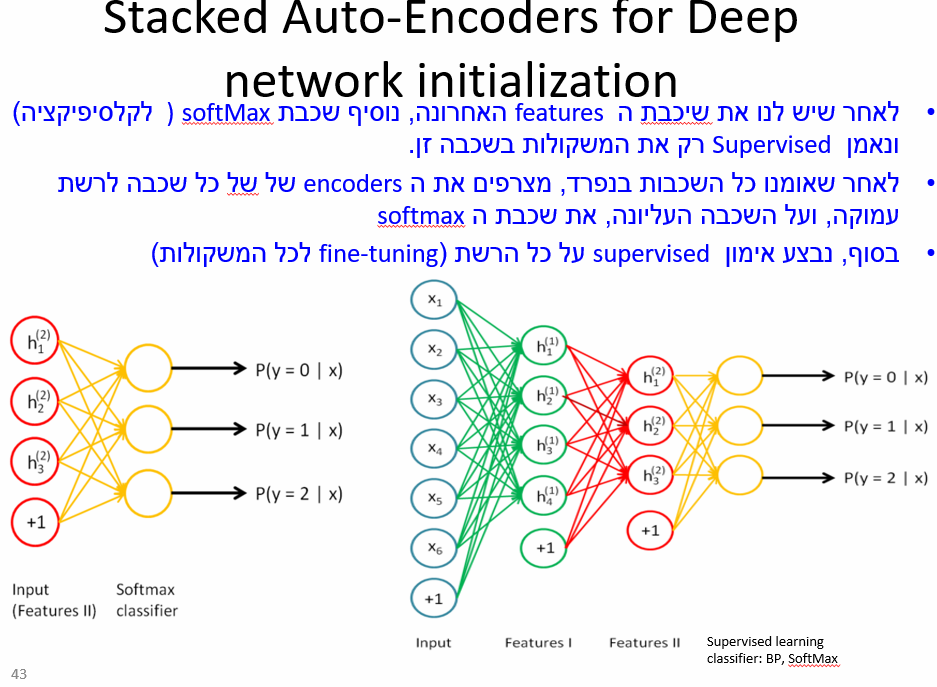
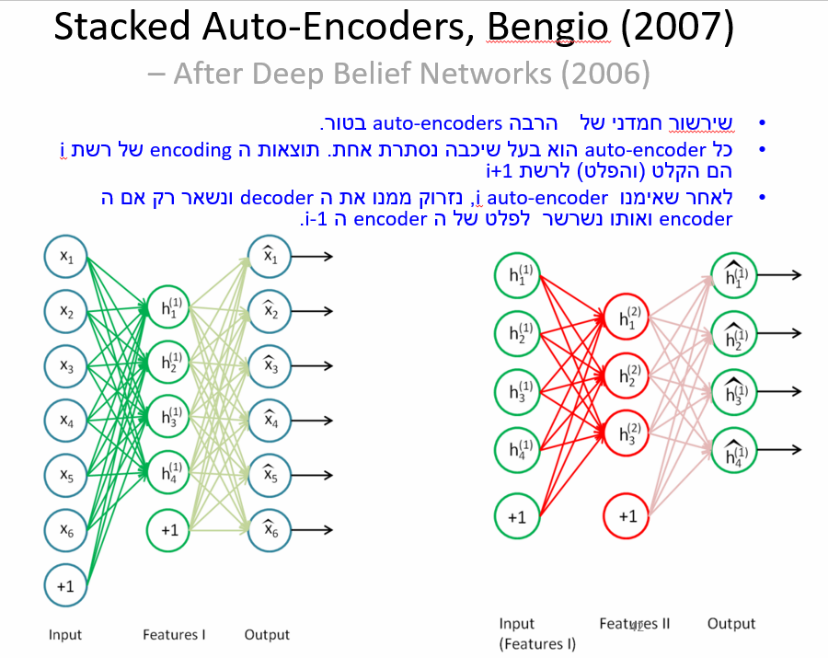
****

****



****

****

****

**פרספטרונים:**

**ברשתות נוירונים מלאכותיות, נשתמש במשקל כדי לציין את חוזק הסינפסה**:משקל חיובי משקל שלילי. השפעת נוירון אחד על נוירון שני מבוקרת באמצעות "חוזק" הסינפסה (שלילי או חיובי): המשקולות הסינפטיים משתנים ומתאימים את עצמם .הרשת כולה לומדת לעשות חישובים שימושיים: זיהוי עצמים, הבנת שפה, תיכנון, שליטה על הגוף.

**בניה הנדסית של רשתות מלאכותיות:**

הנוירונים בשכבות הקלט והפלט יקראו **Visible units**

יותר משכבה חבויה אחת**: רשת עמוקה**

ללא שכבות חבויות בכלל: רשת מוגבלת חישובית.

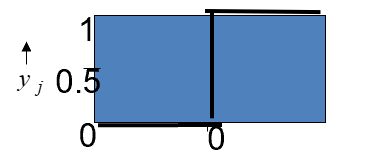
**סוגי נוירונים :**

**Z (weighted sum) -**  **Activation -**

כאשר g הוא פונקצית אקטיבציה.

**BTU- Binary Threshold Unit :**

קלט של פונקציית אקטיבציה זו הוא בין 0 ל-1.





**Logistic neurons(sigmoid) –**

פלט:מספר ממשי בין 0 ל-1

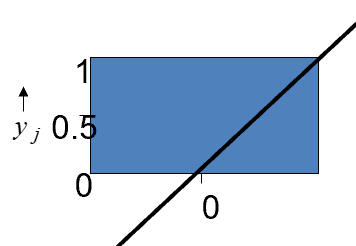
כאשר הסכום z הוא אפס הפלט יהיה 0.5 .

**כאשר הסכום z הוא שלילי הפלט יהיה מתחת לחצי.**

**כאשר הסכום z הוא חיובי הפלט יהיה גדול מחצי.**

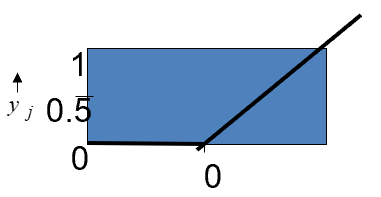
בכדי לקרב את ה-sigmoid ל- BTU נוסיף קבוע טמפרטורה:

* כאשר t=1 , g היא הפונקציה הלוגיסטית הרגילה
* כאשר t מתקרב ל 0, g מתקרב ל BTU
* כאשר t גדול, הפונקציה משתטחת
* נקבל:



**Linear Unit –**

סכום הקלטים המשוקלל פונקציית זהות.

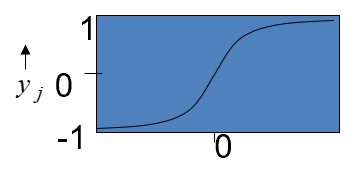


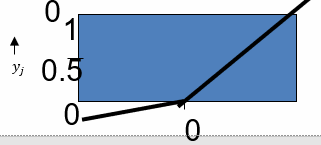
**Rectified-Linear Unit (RelU) –**

פלט הקטיבציה יהיה z אם חיובי, 0 אם שלילי.

**Hyperbolic Tg neuron –**

פלט הקטיבציה הוא בין 1- ל-1.



****

**Leaky RelU –**

הוא שבר [0,1)

אם 0, מקבלים RelU רגיל

אם שווה 1, מקבלים נוירון לינארי

ואם שבר , הוא מוכפל ב z רק כאשר z<0

**יתרון: הנגזרת לא מתאפסת כאשר z<0**

**הפרדה לינארית**

במרחב הקלטים ה- n מימדי: כל דוגמת קלט היא נקודה.

וקטור של n+1 משקולות מגדיר היפר מישור n-1 מימדי: .

כל היפר-מישור מפריד לינארית: יש צד חיובי וצד שלילי ,בהתאם לסימן של המכפלה WX:

🡸 '+'

🡸 '-'

פעולת הנוירון היא הפרדה לינארית בעזרת היפר מישור.

לדוגמא: אם הקלט היה ממימד 2 אז מספר המשקולות הוא 3 וההיפר מישור שהיה מפריד ביניהם היה ממעלה 1 (מימד 1).כלומר, ההפרדה היא בעזרת p קו ישר:

**קיימים מגבלות על הפרדה הלינארית כמו למשל בXOR.**

**הוכחה:**

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

w1+w2 >-2b

וגם

w1+w2 ≤-2b

סתירה!!

b <0

w2 ≥-b

w1 ≥ -b

w1+w2 < -b

0w1+0w2+b <0

0w1+1w2+b ≥0

1w1+0w2+b ≥ 0

1w1+1w2+b < 0

**מגבלות הפרדה**

יש מגבלות חמורות על מה שניתן לבצע בעזרת BTU בודד ללא שכבה נסתרת:  
לא ניתן לממש בעזרת BTU בעיות מעשיות פשוטות כמו: ,Parity,זיהוי תבנית שזזה.אך בכל זאת שימושיים למשל:כאשר הנתונים מעטים ורועשים, הפרדה לינארית היא לגיטימית ועשויה דווקא לשפר את יכולת ההכללה. בנוסף, קימות טרנספורמציות לקלטים שהופכים קבוצות קלטים בלתי ניתנת להפרדה לינארית, לכזו שכן ניתנת להפרדה לינארית (בד"כ במימד גבוה יותר).

לעיתים (קרובות) אנחנו זקוקים להפרדה "עדינה" יותר:נצטרך להוסיף נוירונים נסתרים.ככל שהרשת תהיה גדולה יותר, גבול ההחלטה יוכל להיות עקום ולהתאים את עצמו לנתונים.

**הפתרון למגבלות של נוירון הBTU יכול להיפתר ע"י BTU רב שכבתי** - בעזרת שכבה נוספת ניתן להפריד כל פונקציה, קיים אלגוריתם בנייה שמוכיח את הטענה, אך בכל זאת קיים חיסרון ,עבור קלט ממימד n, נצטרך מספר אקספוננציאלי של features לא מעשי עבור בעיות פרקטיות, מספר אקספוננציאלי של Features שלעיתים לא צריך מספר רב של נוירונים נסתרים, והרשת לא תכליל.

**משפט הבנייה לפספטרונים:**

בנית feature לכל שורה בטבלת האמת: לכל נוירון נסתר מתאימה תבנית אחת ורק אחת

נפריד בין שורות האמת המסווגות כ-1 לשורות האמת המסווגות כ-0.

**נוירון הרגיש בדיוק לתבנית מסוימת אחת:**

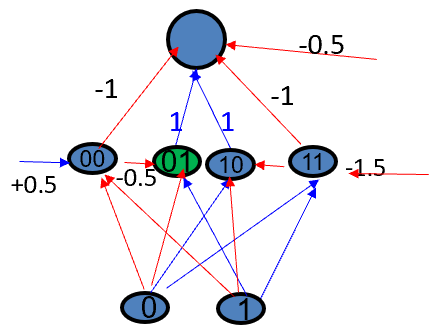
כאשר יש m קלטים שצריכים להיות 1 ו k קלטים שצריכים להיות 0,

משקל חיובי +1, לכל אחד מהקלטים שצריכים להיות 1

משקל שלילי -1 , לכל אחד מהקלטים שצריכים להיות 0

הסף bias : קצת מתחת ל m- (מספר האחדים( בדר"כ בחצי.

לדוגמא: בשער AND צריך להיות רגישות ל-11 לכן המשקולות יהיו 1 והbias יהיה -1.5



**לדוגמא בניית XOR:**

**למידה של פרספטרון**

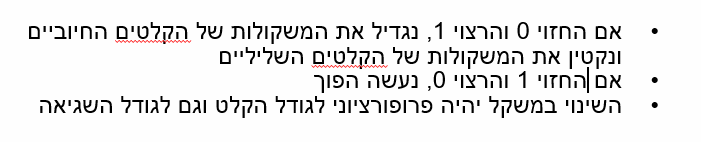
* + יחידות הקלט (features) יכולים להכיל ערכים ממשיים,יחידות הפלט בינאריות.
  + יחידות הקלט מחוברים לכל נוירוני הפלט Fully connected, המשקולות מהקלטים לפלטים נלמדות במהלך האימון.

הארכיטקטורה:

* + Feed Forward-הסיגנלים של הירי מתקדמים רק קדימה,ואין לולאות עצמיות(קשרים),Recurrent.
  + מטריצת משקולות (**מספר השורות כמספר הפלטים, עמודות כמספר הקלטים**):  
     wij המשקל בין קלט i לפלט j

**אלגוריתם למידת הפרספטרון:**

**המטרה:** בהנתן קבוצת נקודות במרחב הקלט עם תווית +/-. רוצים להפריד בינהם ע"י היפר-מישור.   
ז.א. רוצים למצוא את משקולות הקשרים W שיצרו מישור מפריד: w1x1+w2x2+… wnxn+w0=0.

1. נתונה רשת של פרספטרון בודד וקבוצת אימון:
   * קבוצת האימון מכילה K תבניות מסווגות :{}
   * תויות
   * וקטורי הקלט הם
2. התחל ממשקלות אקראיים מכל יחידת קלט אל יחידת פלט:  
   עבור על כל התבניות (k=1…K) שוב ושוב ,  
    חשב שגיאה ועל פיה שנה את המשקולות , עד שמעבר על כל התבניות אינו גורם לשינוי משקלות:
   * עבור תבנית חשב את פלט הפרספטרון **(מצב מצוי**):
   * חשב את השגיאה:
   * עדכן את וקטור המשקלות עבור כל סינפסה שמחברת בין יחידת קלט יחידת הפלט:

*– קצב למידה.*

במקרה ולא תיהיה הפרדה לינארית מושלמת הפרצדורה לא תעצור אף פעם כי היא כל הזמן תנסה לשפר את המשקולות. הפתרון הוא להגביל את ריצת הלולאה ע"י הוספת תנאים כמו למשל, עצירה כאשר שגיאת הקלסיפקציה (\*ErrRate) מספיק קטן . ErrRate =

**סוגי למידת מכונה:**

***Supervised: Labeled examples***

מניחים קיום של מורה: Labeled Data

**מתודלוגיה ב-3 צעדים:**

1. אימון-Training - למידה מתוך קבוצת דוגמאות אימון על מנת למזער את השגיאות. דוגמת אימון כוללות גם פלט "רצוי" שנתן המורה, Labeled Data. בסיום הלמידה, מייצרים מודלh שאותו ניתן לבדוק מול קבוצת דוגמאות בוחן .
2. ולידציה-validating – הערכת המודל ע"י מידע שלא ראה עוד, קבוצת דוגמאות בוחן Test Data.
3. מבחן סופי- שימוש במודל באפליקציות אמיתיות.

הבעיה בלמידת מכונה מונחית היא להסיק מתוך נתונים מתויגים, היפותזה המקרבת פונקציה (ההפרדה המושלמת אשר איננה ידועה) הממפה במציאות קלטים לפלטים,כך ששגיאת החיזוי lossT(h) על דוגמאות הבוחן(טסט) Tתהיה מזערית. רוצים שההיפותזה המקרבת תהיה מסוגלת להכליל ולחזות פלטים נכונים מתוך קלטים חדשים (שאף פעם לא הראינו למכונה).יכולת ההכללה נמדדת על דוגמאות מבחן שלא ניצפו בזמן האימון.

***Unsupervised: Unlabeled examples***

Data Unlabeled. המטרה למצוא סדר בנתונים: למצוא חריגים או להפיק מאפיינים חשובים

**ברוב אלגוריתמי הלמידה, מבצעים אופטימיזציה של פונקציית עלות כלשהי על מנת למצוא מודל שממזער את העלות.** **מערכות לומדות: לא שומרות את הנתונים בזיכרון, לא משננות, אלא, מכלילות, מזהות תבניות ואז יודעות להגיב נכון על נתונים דומים.**

**רגרסיה לינארית ולוגיסטית**

* **רגרסיה: הפלט של המודל הוא נומרי**
* **קלסיפיקציה: הפלט של המודל הוא אוסף קטגוריות (והסתברויות)**
* **נוירון לינארי-**יחידת חישוב שמחשבת היפותזה לינארית, פונקציית האקטיבציה הינה לינארית (פונקצית הזהות).

**רגרסיה לינארית במשתנה בודד (Single variable linear regression)**

נתון אוסף נקודות {,} . נרצה למצוא קו ישר המתאר את הנתונים בצורה הטובה ביותר. נרצה למצוא שני פרמטרים: כך שהמשוואה תהיה משוואה של קו ישר המתאים ביותר לאוסף הנקודות.

המטרה, ליצר מודל y=h(x) אשר ממזער את שגיאת ההכללה על דוגמאות המבחן ב T כדי למצוא את h(x),

ה-Learner מנסה למזער את שגיאת האימון ב D.

**GD למיזעור פונקצית lossD(w) דיפרנציאבילית עבור מודל hw(x)**

**GD ,SGD**

GD - כל צעד (ממוצע גראדיינטים של כל דוגמאות האימון): נתקדם במאונך לקו הגובהGD , הוא אלגוריתם off-line,משתמשים בכל ה training set (Epoch) כדי לבצע צעד אחד ב Batchאם קבוצת האימון גדולה, צריך למצע המוני גראדיינטים לפני שעושים צעד בודד.

SGD - מבצעת קירוב לגראדיינט האמיתי, מכיוון שכל דוגמא גורמת לשינוי ב W וחישוב הגראדיינט בדוגמא הבאה ישתנה. העלות תיטה לרדת באופן תזזיתי עד שנסתובב בקרבת נקודת המינימום.

**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, התזזתיות עוזרת לפעמים לברוח ממינימום מקומי
* ניתן להשתמש ב on-line learning   
  (המשך הלמידה גם במצב production/test – בד"כ עם קצב למידה גבוה)

**חסרונות:**

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

GD מבצע עידכון לאחר שעבר על כל הדוגמאות כלומר הוא מחשב את הגרדיאנטים לאחר כל דוגמא ואז מבצע ממוצע וכך הוא מחשב את ואז מבצע עדכון משקולות. SGDלעומת זאת משפר לאחר כל דוגמא, כלומר, הוא מחשב לאחר כל דוגמא את ואז ממבצע עדכון של המשקולות.

**SGD Mini-Batch**

בחירת מספר m (<200) של דוגמאות אקראיות כדי לחשב קירוב לגראדיינט , כלומר הוא מחשב את הגרדיאנטים עבור כל minibatch ולאח מכן מחשב את ממוצע הגרדיאנטים עבור כל הדגומאות ומעדכן את המשקולות.

minibatch בערך 50-200 דוגמאות, טוב יותר מSGD וגם הרבה יותר מהיר (פחות חישוב),השוני בדגימה בין ה minibatches מייצר סטוכסטיות מסוימת שמשפרת את היכולת למצוא מינימום גלובאלי לשגיאה.

**מתי לעצור ב GD/SGD שימוש בוואלידציה**

נפריד את האימון ל2 קבוצות: קבוצת אימון וקבוצת ואלידציה (קבוצת דוגמאות נפרדת ושונה מקבוצות האימון והטסט) .ההוואלידציה היא כמו טסט לאימונים (אך איננה המבחן הסופי)ק.לכן ,קריטריון עצירה יהיה כאשר שגיאת הוולידציה נמוכה ואינה משתפרת.

***N-fold Cross Validation:***

נחלק את ה training ל N קבוצות,נשמור 1/N מהנתונים כקבוצת ולידציה ואז נאמן על (N-1)/N מהנתונים ונבדוק על הוולידציה.נצטרך לבצע N אימונים שונים בכל פעם על קבוצת וולידציה אחרת ולאחר מכן נקבל N הערכות שונות ל validation loss.בד"כ נבחר במודל עם הביצועים הטובים ביותר אך אפשר גם לצרף את המודלים הטובים לEnsemble.

***קונבנצית האינדקסים והכתיב ווקטורי***

* *yi(k) האקטיבציה של נוירון i בשכבה k y (1)=x. האקטיבציה של שיכבה 1 היא הקלטים עצמם/*
* *wk  מטריצת משקולות המחברת בין שיכבה k לשכבה k+1.   
   שורה i במטריצה כוללת את כל המשקולות המחוברות לנוירון i בשכבה הבאה.*
* *wki,j המשקל של הסינפסה המחברת את הנוירון ה j משיכבה k לנוירון i בשכבה k+1*
* *כקונבנציה: מוסיפים נוירון bias לכל שיכבה: 1=x0 בקלט ו y0 (k) בשכבה k*
* *בכתיב ווקטורי עבור השיכבה ה kית: k=2…..L*

*z(k)=w(k-1) y(k-1) ו- y(k)=g(z(k))*

*Z (2) =w1x סכום הקלטים המשוקלל בשכבה הנסתרת הראשונה*

*y (2) =g(z (2)) האקטיבציה של השכבה הנסתרת הראשונה*

***פונקציות עלות :***

***Cross-Entropy -CE***

**MSE-Mean Squared Error**

**הערה:לעיתים מחלקים ב-2 כדי להפתר מה-2 שכופל את הנגזרת.**

**כלל העידכון המשקולות**

עידכון המשקולות צריך להיעשות סימולטנית בכל המשקולות: קודם כל יש לחשב את הנגזרת החלקית לכל משקל. ורק אחרי שכל הנגזרות החלקיות חושבו יש לעדכן את המשקולות. אחרת (אם העידכון הוא סידרתי ולא סימולטני) שינוי משקל אחד עלול להשפיע על חישוב הנגזרת החלקית למשקל השני... האלגוריתם יעבוד, אך יתנהג בצורה שונה מהאלגוריתם הרגיל (כשאומרים grad descent, מתכוונים לעידכון הסימולטני).

השינוי במשקולת הוא מינוס הנגזרת החלקית כפול קבוע קצב למידה .

**- נגזרת של פונקצית האקטיבציה לפי Z.**

**– נגזרת פונקציית ה-loss לפי Y.**

**– נגזרת Z לפי W.**

**אלגוריתם ה Back Prop כחלק מ SGD-**

1. **קבע את ערכי המשקולות ברשת לערכים אקראיים קטנים  
   עבור שוב ושוב על הדוגמאות בקבוצת האימון D (עד לתנאי עצירה).**
2. **בחר דוגמא p מקבוצת האימון ואתחל בעזרתה את ערכי הקלט x=y(1) (clamp)**
3. **Feed Forward: לקבלת ערכי כל הנוירונים: k=2…L**
4. **BP - חשב את ערכי הדלטות:  
   התחל בחישוב דלטה בשיכבה האחרונה:**

**המשך אחורנית שיכבה אחר שיכבה: k=L-1,,,3,2**

1. **חשב את השינוי הדרוש במשקולות לכל משקולת :**

**עדכן משקולות: כל Minibatch חשב ממוצע של השינויים של m דוגמאות ועדכן משקולות:**

***חישוב דלתא עבור דוגמא בודדת:***

**עבור נויורנים לוגיסטים:**

**דלתא בשכבה האחרונה:**

**Cross Entropy:**

**MSE:**

**דלתא בשאר השכבות:**

**עדכון משקולות:**

**קצב למידה**

אם רואים כי השגיאה עולה בהתמדה או לא יורדת, יתכן והקצב גבוה מידי והעידכון זורק לכיוון השני של המינימום לנקודה גבוהה יותר,לכן כדאי לנסות להוריד את הקצב.מצד שני, אם הקצב נמוך מידי, תהיה התקדמות איטית מידי. בכדי שנוכל להחליט על קצב למידה אופטימלי נבדוק מול קבוצת וולידציה של מספר קצבי למידה שונים, ונבחר ב LR הגדול ביותר שאינו מרע משמעותית את הוולידציה (validation loss).

**המשוואה הנורמלית:**

: NORMAL EQUATION נוסחה מפורשת (אנליטית) למציאת המשקולות ברגרסיה לינארית שממזערים MSE.

**למידת מכונה: בעיות ופתרונות**

**בעיית מינימום מקומי - Problems with Local Minima**

BP הוא חיפוש חמדני המבוסס על Gradient Descent, משטח השגיאה נראה כמו הרים ועמקים פונקצית העלות איננה קמורה ויתכנו מספר נקודות מינימום מקומיות.איתחול שונה של הרשת, עלול להביא לנקודת מינימום רדודה.שימוש ב SGD עוזר במקצת : התזזיתיות מאפשרת אקראיות.

**בעיית הגראדיינטים הנעלמים בנוירונים לוגיסטים**

הביטוי y(1-y) מופיע בחישוב הדלטא בכל השכבות.במהלך החילחול אחורה מוכפלים הביטויים הללו זה בזה. מספיק שנוירון אחד יקבל ערך קרוב ל 0 או 1, הגראדיינט יעלם.אחד הפיתרונות, למצוא אקטיבציות שהנגזרת הפנימית שלהם אינה מתאפסת לרוב.

הערה : ל hyperbolic tag , יש אותה הנגזרת כמו לסיגמויד.

קיימים פונקציות שלא תמיד מתאפסות כדי לפתור את הבעיה הזו כמו Relu,Leaky Relu.

ה-Leaky Relu בא לפתוראת הבעיה של ה-Relu שחלחול לאחור הוא נעצר כי הנגזרת התאפסה. הנגזרת היא 1 אם z>0 ו 0 אחרת.הביטוי (y)(1-y) אינו קיים ,אבל אם Z<0 הנוירון משתתק ,הנגזרת מתאפסת וחילחול הגראדיינט אחורנית נעצר.

**מומנטום: להתגבר על מכשולים קטנים בעזרת מהירות גבוהה**

בהתחלה הוא מנצל את הגראדינט, אך ברגע שצובר תאוצה הוא לא יורד במורד הכי תלול. המומנט שלו גורם לו להמשיך בכיוון הקודם שלו,מתעלם מעקמומיות קלה ובונה מהירות בכיוונים שיש בהם גראדיינט עדין אבל קונסיסטנטי.

***Δ***

**או**

* קבוע **β** שומר חלקית על השינוי הישן (בד"כ מספר בין 0 ל 1)
* השינוי הנוכחי במשקל: חלק **β** מהשינוי הקודם + הגראדיינט הנוכחי

כשהשגיאה יורדת בהתמדה, המהירות גוברת,שק**ול להגברת קצב הלמידה,**המומנטום מפלטר שינויים קטנים בתדר גבוה במרחב השגיאה אך קיימת בעיה נוספת: עוד פארמטר שצריך לבחור.

**Overfitting: שגיאות high-variance**

הדוגמאות בקבוצת האמון כוללות מידע על תבניות וחוקים, אבל גם כוללות רעש ועלולה להיווצר ,טעות דגימה sampling err) ). זיהוי שגוי של תבניות רנדומליות שנובעות רק מהבחירה של הדוגמאות לקבוצת האימון.כאשר מתאימים מודל, אין דרך לזהות מי מהתבניות היא תבנית "אמיתית" ומי רק נגרמה בגלל מדגם.אם המודל מאוד גמיש, הוא יכול למדל את שגיאת הדגימה מאוד טוב וזה ייצור שגיאה גדולה בולידציה.

**הפתרונות:**

**הכבדה: נקשה על הלמידה. המודל יתמקד בתבניות ה"אמיתיות"**

* **נוסיף Data**  הרשת תתקשה להתאים עצמה בדיוק לקבוצת אימון גדולה.בנוסף אפשר גם להזריק רעש לדוגמאות(לטשט תמונות ,להפוך אותם וכו).
* **רגולריזציה:**

**הענשה של משקולות גדולים-רגולריזצית משקולות**

הענשת כל המשקולות, גורמת להעדפת היפותזות פשוטות יותר ז.א פונקציות רגרסיה (או גבולות החלטה) "חלקות" יותר. איננו יודעים על איזו משקולת לוותר ולכן נעניש את כולם(חוץ מהביאס).

כלל עידכון המשקולות כאשר "מענישים" משקולות מתאפיין ב"שיכחה"

**ההשפעה של "שיכחה" של משקלות**

מונעת מהרשת להשתמש במשקלים שלא צריך אותם היא תצליח להכליל כי היא מונעת מהרשת להתאים עצמה לטעויות דגימה.המשקל יהיה גדול רק אם השפעתו גדולה על השגיאה המקורית.

המודל המתקבל הינו "חלק" יותר: בגלל שהמשקלים קטנים, הפלט משתנה לאט יותר כאשר הקלט משתנה אם לרשת יש 2 קלטים דומים, תעדיף לשים רק חצי מהמשקל על כל אחד במקום כל המשקל על אחד .  
פחות הסתמכות על קלטים בודדים- יותר החלטה קולקטיבית של הרבה features

* **עצירת האימון לפני שהרשת תספיק ללמוד "יותר מידי"**

מתחילים ממשקלות קטנים ונותנים להם לגדול עד **שהביצועים על ה validation set** מפסיקים להשתפר ואז עוצרים וחוזרים למשקולות בהן שגיאת הוולידציה היתה מינימלית.כאשר המשקולות קטנים, הפונקציה הלוגיסטית נמצאת בטווח שהיא דומה לנוירון לינארי,רשת כזו מתנהגת כאילו היתה לינארית (רשת מוגבלת) High-bias ,אין בה יותר יכולות מרשת לינארית שבה הקלטים מחוברים לפלטים ישירות. לפיכך, בהתחלה הרשת עושה under fitting.כאשר המשקולות גדלות בהדרגה, הנוירונים מתחילים להשתמש בטווחים הלא לינאריים ,יכולת הרשת להתאים את עצמה ל data גדלה בהדרגה.ברגע שהשגיאה לא משתפרת, מפסיקים ללמוד כך לא נותנים מספיק זמן למודל להתאים עצמו גם לתבניות האקראיות והפחות משמעותיות, שנוצרו בגלל שגיאת הדגימה.

**יש להיזהר ולא לעצור לפני הזמן**.יתכנו גליצ'ים רועשים שיכולים לגרום לשגיאה זמנית גדולה בוולידציה.זה יכול להטעות אותנו לעצור מוקדם מידי כדאי לוודא ששגיאת הוולידציה לא יורדת במשך מספר איפוקים

* **Ensembles - חכמת ההמון, צירוף של הרבה מודלים שונים.**

כאשר כמות ה Data מוגבלת, והמודל גמיש מידי, עלולים לקבל Overfitting: (low bias, high variance)  
אם יש ברשותנו מודלים כאלו אך הם שונים אחד מהשני ונותנים תחזיות שונות, ממצעים את ההסתברויות הנחזות או אם מעונינים בהחלטה משתמשים בהצבעת הרוב (עבור רגרסיה ממצעים את התחזיות)

הסבר השגיאה מורכבת מ שגיאות bias שנובע מחוסר יכולות של המודל ומשגיאות variance שנובע מהתאמה לשגיאת דגימה ע"י מיצוע של מודלים עם Bias נמוך ו-variance גבוה, אנחנו מעלימים את ה-variance.

נוכל ליצור אנסמבלים שונים ע"י שינוי ה-Training set,מאמנים מודלים שונים על תת קבוצות שונות של הdata  
למשל תוך שימוש ב N-fold cross validation נקרא Bagging. את הפלטים ממצעים (או מבצעים סווג על פי הצבעת הרוב).

Boosting: משתמשים בסידרה של מודלים "חלשים" (ללא יכולת התאמה גבוהה), כל מודל מתרכז בלמידה של המקרים שהמודלים האחרים לא הצליחו לסווג נכון:

משתמשים באותו Data, אך מגדילים את משקל הדוגמה –תדירות הופעתה– במידה והמודלים הקודמים לא סיווגו אותה נכון. מקבלים סידרת מומחים (שכל אחד חלש מידי בשביל Overfitting) שנותנים תחזיות שונות למיקרים קשים. את המודלים השונים שמתקבלים ניתן למצע ע"י הצבעת הרוב / מיצוע חשבוני של הסתברויות/ מיצוע גיאומטרי או לשקלל על פי חשיבות השגיאות שמטופלות ע"י כל אחד.

* **Dropout: רגולריזציה בעזרת הרג נוירונים אקראיים,שיטה יעילה יותר לensemble-**

באימון מקשים על הרשת ללמוד,בכל batch של אימון: 40%-60% מהנוירונים הנסתרים מוסרים רנדומית  
שינוי המשקולות מתבצע רק עבור הנוירונים שנשארו.באיטרציה הבאה, יחזרו הנוירונים לתחיה (עם משקולותיהם שלא עודכנו) וקבוצה אחרת אקראית של נוירונים יוסרו.בזמן הטסט/ וולידציה, כל הנוירונים ישתתפו ברשת אחת.

המטרה איננה משקולות קטנות, אלא יצור זול של המוני רשתות חלשות שעובדו ביחד כדי להכליל טוב.

הקושי: נוירונים לא יכולים "לסמוך" על vנוירונים אחרים מכיוון שבאיטרציה הבאה הם לא יהיו בכל איטרציה, יש רשת קטנה שתתקשה להתאים את עצמה.

**אתחול המשקולות**

אם כל היחידות מתחילות במשקולות זהים, לא ניתן ליצור features שונים, יש לשבור את הסימטריה ע"י הגרלת משקולות רנדומיים.אחרת, ילמדו משקולות זהות (לכל הנוירונים המחוברים לאותו קלט)ניתן לבצע reinitialize למשקולות ברגע שמגיעים למצב שהשגיאה אינה יורדת (אולי נתקענו במינימום מקומי).דבר זה יכול לפתור את בעיית ה-overfitting. אתחול משקולות קטנים כדי שנתחיל עם מודל חלש (underfitted) אשר, רק בהדרגה ישפר את התאמתו.

**מדידות מודלים שונים KPI (err, accuracy, Precision, Recall, F, confusion matrix)**

**Accuracy:** מיזעור טעויות מכל הסוגים (FT+FN) מתוך כל המיקרים

**: Precision** מיזעור false positives, מתוך המסווגים כ positives.

**: Recall** מיזעור False Negatives "פיספוסים" מתוך אלו שאמורים להיות מאותרים (החולים בפועל)

**Specificity:** מזעור False Positives מתוך סך השליליים בפועל t=0

**S**

**f-measure -ממוצע הרמוני:** נועד כדי להשוות בין אלגוריתמים שונים.

**ארכיטקטורות עמוקות:**

הרעיון מאחורי ארכיטקטורות עמוקות הוא שבכל שכבה נוכל לקבל איזשהו ייצוג פנימי, המאפינים הבסיסים שמתקבלים מבחינים בין צורות פשוטות ,ממאפיינים "נמוכים" ניבנים מאפינים גבוהים יותר, מהמאפינים "הגבוהים " נבנים מאפינים אחרים וכך הלאה

**Auto-encoders**

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, יש קושי בהתאמה מדויקת לנתונים וכך מתגלים features החשובים (ייצוג פנימי). אין צורך במורה, ניתן לקחת תמונות (או הקלטות קול) ללא תוויות (או ללא תמלול( וללמוד להוציא מהם את המאפיינים החשובים שיאפשרו שיחזור של הקלט לאחר דחיסה.

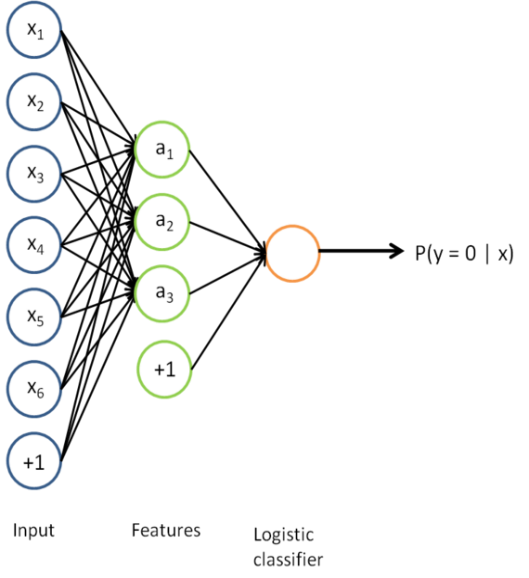
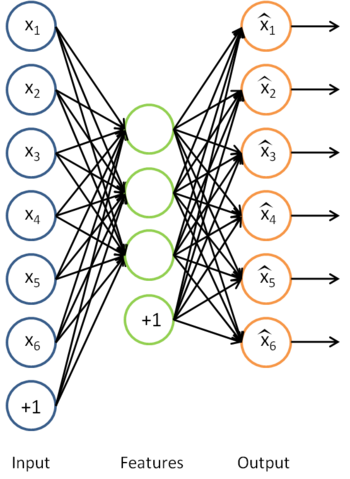
**קיימות מספר שיטות ליצירת צוואר בקבוק**

**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)

**דלילות (Sparsity)** – מספר הממדים (אורך הווקטור) הדרוש לייצג מילה בודדת הוא כאורך מספר המילים הקיימות בכלל גוף הטקסט (corups) העלול בקלות להגיע לעשרות אלפי מילים ויותר.ברור כי ייצוג זה מאוד לא חסכוני ודורש משאבי מחשוב משמעותיים ע”מ להזינם למערכת לומדת.

**למידה-semi-supervised**

****תחילה עושים למידה unsupervised , למציאת יצוג פנימי הכולל features כלליים וטובים בגלל צוואר בקבוק בשכבות הנסתרות (דחיסה), אין שינון של המידע אלא מתגלים features חשובים.לאחר האימון לעשה למידה supervised על ה-features שהתגלו.

צעד 1:למידה unsupervised במקרה זה שכבת הנוריונים מחוברים לencoder- ו-decoder

צעד 2: חיבור הנורונים לנוירונן פלט אחד.ואז למידה supervised

**Training Layer by layer**

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.

**Stacked Autoencoders-שיטה לאתחול משקולות לקלסיפיקציה עמוקה:**

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.לאחר שיש לנו את שיכבת ה features האחרונה, נוסיף שכבת SoftMax ) לקלסיפיקציה)ונאמן Supervised רק את המשקולות בשכבה זו.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)

**רשתות קונבולוציה**

רשת עמוקה הבנויה משכבות רבות מיישמת את הרעיון של קישוריות לוקאלית: מעט משקולות, המשותפים לאיזורים שונים וחופפים בתמונה.ברשת יש פילטרים שהם feature detectors שעובדים על קלט לוקאלי, שיתוף משקולות, אותו פילטר עובד על חלונות שונים - אך חופפים.

פילטר אחד מופעל על הרבה חלונות שונים אך חופפים ומייצר תמונה מפולטרת (Convolved) דו מימדית הנקראת גם Activation-map, שכבה נסתרת של נוירונים שהאקטיבציה שלהם מיצגת "פיקסלים" של תמונה מפולטרת. המשקולות של אותו נוירון, משותפות לכל הנוירונים בתמונה המפולטרת.

על כל חלון, מסתכלים כמה סוגים של פילטרים ונותנים מימד שלישי out channels.K פילטרים, מיצרים k תמונות(ערוצים).ברשתות אלה יש שימוש בשכבות Fully Connected שמשמשים כשכבות עליונות לפני שכבת ה SoftMax.בנוסף עושים רגולריזציה כמו dropout,L2,momentum.

**ההיגיון מאחורי שיתוף המשקולות:**אם הצלחנו לזהות features של "חתול" בצד שמאל של התמונה, המשקולות יועתקו לכל חלקי התמונה.הfeatures- יהיו זמינים גם אם לא ראינו חתולים בכל הפוזיציות בתמונה.לא נצטרך ללמוד שוב את אותם features בכל מקום אחר של התמונה.זה עדיין לא נותן לנו translation invariance: נצטרך שכבות גבוהות יותר שילמדו שחתול שאותר משמאל וחתול שאותר מימין, זה חתול.

**שיכבת קונוולוציה**: היחידות הנסתרות הם Features הרגישים לסביבה לוקאלית (חלון קטן) ונקראים גם Detectors/ filters ,פילטרים מסוג מסוים, הינם רגישים לסביבות לוקאליות שונות אך משתפים משקולות . בזמן האימון, מחשבים את השינוי במשקולות של כל פילטר בנפרד ואז ממצעים את השינויים ושומרים סט יחיד של משקולות לכל סוג פילטר.בשכבות קונוולוציה משתמשים בד"כ באקטיבציות RelU.

**מספר הנוירונים בשכבה אחת:**

* Accepts a volume of size קלט:
* Requires four hyperparameters:
  + Number of filters
  + their spatial extent ,(FxF מימדי מטריצת הקונבלוציה)
  + the stride  ,
  + the amount of zero padding .
  + padding = same שומר על מימדי התמונה לאחר פילטור
  + padding = same =
* Produces a volume of size where:
  + W2 =
  + H2 =
  + D2=K

**מספר המשקולות עבור נוירון בודד בשכבה:** (

**מספר סינפסות:**

**Pooling**

מטרתה בד"כ: לדגום את השכבה הקודמת ולהוריד את המימד (אך לשמור את האינפורמציה הבולטת).

Pooling מועיל לטובת translation invariance: למשל ע"י מיצוע (או max) של 4 features שנמצאים בסמוך.

Pooling גם מוריד מימד, מה שמאפשר (מבחינה חישובית) להוסיף עוד סוגי פילטרים.

Max Pooling: מסתכל על חלונות שונים ודוגם את הערך המקסימלי בכל חלון

חיסרון: אחרי מספר רמות של מיצוע, מאבדים ידע על המיקום האמיתי ומתקשים בזיהוי יחסים גבוהים יותר בין חלקי האוביקט.

Average Pooling: מסתכל על חלונות שונים ודוגם את הערך הממוצע בכל חלון

**תוצאה לאחר Pooling:**

* Accepts a volume of size
* Requires two hyperparameters:
  + their spatial extent F,
  + the stride S
* Produces a volume of size  where:
* W2 =
* H2 =
* D2=D1

**Microsoft Asia Residual net**

חוקרים הבחינו כי העמקת רשתות שעבדו היטב גורמת להרעה בשגיאות. היו הרבה נתונים, ושיטות רגולריזציה לא הועילו(ז.א. לא תמיד הייתה בעיה של overfitting).הסיבה לכך הייתה בעיית הגאדיאנטים הנעלמים.

הפתרון:מעקף שמאפשר לרשת ללמוד בקלות ששכבה מסוימת היא מיותרת (בקלות ניתן לגרום למשקולות לאפס את ה RelU, ולאפשר רק **פונקציית זהות** בין השכבות.אם כך, רשת עמוקה יותר עם מעקפים תצליח להגיע לביצועים של רשתות טובות ורדדות ממנה.ואולי אפילו יותר טובים, בגלל היררכית features עמוקה יותר.

**HighWay Networks - שיטה כללית למעקפים**

בכל שכבה ישנם שערים ששולטים על פלט השיכבה:

* קלט x נכנס לשכבה לא לינארית והמצב החדש הוא y=g(wx+b)
* הקלט x עושה גם מעקף נשלט שער לוגיסטי אשר מחליט כמה מהקלט x יועבר לפלט של השכבה וישוקלל עם המצב החדש
* הקלט יכול לעבור ישר לפלט בדרך ראשית מבלי לעבור דרך טרנספורמציה לא לינארית נוספת
* למרות שמתווספים משקולות, לא משפיע על ה overfitting.תיאורתית מוסבר: הגראדיינטים נעלמים פחות.

**הסבר:**

חישוב שכבה לא לינארית h:

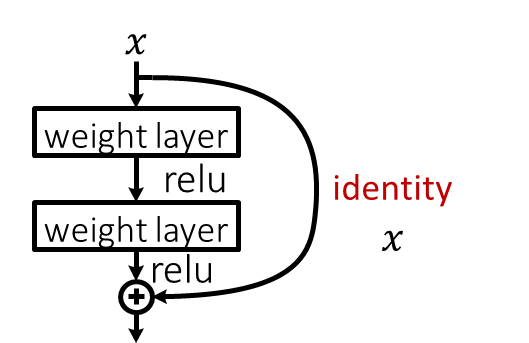
Simple non-linear layer: e.g. g=tanh, relu

חישוב שכבה של שערים t:

Transform using logistic Gate:

t מספר בין (0,1) קובע כמה להעביר הלאה מהנוירון הנסתר החדש (לשכוח את הקלט), (1-t) קובע כמה להעביר את הקלט.

y: ערך השכבה הנסתרת היא אינטרפולציה של h עם הקלט x בעזרת t

אם t=0 , הקלט יועבר הלאה כמו שהוא

אם t=1, y=h שכבה נסתרת רגילה

L1(x)=RelU(w0x + b0)

F(x)= RelU(w1L1(x)+b1)

𝐻 (x)=F(x) + x

**Recurrent Neural Networks (RNN)**

מערכות לומדות קלאסיות מתקשות לרוב בטיפול ברצפים התלויים בסדר ובזמן. ועל כן כבר בשנות ה 80 פותחה מחלקה במשפחת אלגוריתמי ה NN הנקראת Recurrent Neural Network, המיועדת להתמודד עם בעיה זו.

רשתתות RNN דומים לרשתות נוירונים רגילות עם הבדל אחד מרכזי – הפלט של כל שכבה משמש גם כקלט של אותה השכבה לצעד הבא. ארכיטקטורת זו מאפשרת לרשת ל’זכור’ את המידע מהצעד הקודם (ושלמעשה הצטבר עד כה) ובכך לאפשר תיאור של רצפים.

**ארכיטקטורה:**

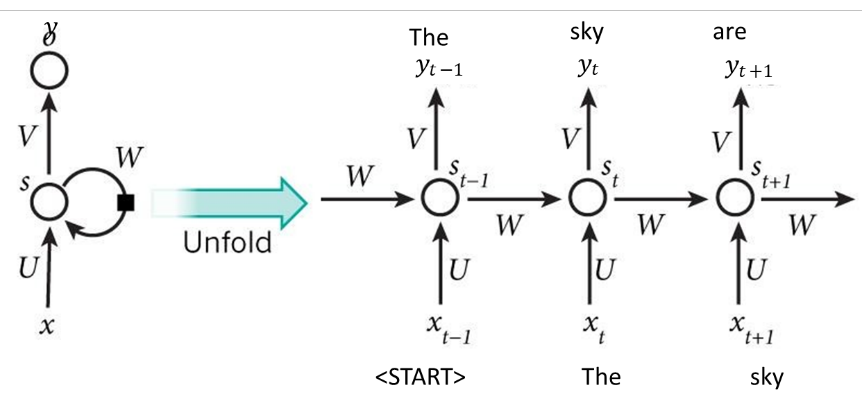
U: מטריצת המשקולות שמחברת קלט Xt בזמן t לשכבה הנסתרת s

W: מטריצת משקולות שמחברת את השכבה הנסתרת בזמן t-1 לשכבה s

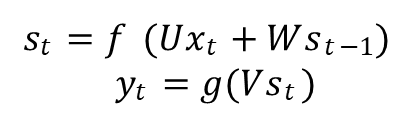
f (): פונקצית אקטיבציה לא לינארית (כדוגמת RelU).

V: מטריצת משקולות שמחברת את השכבה s לשכבת הפלט y

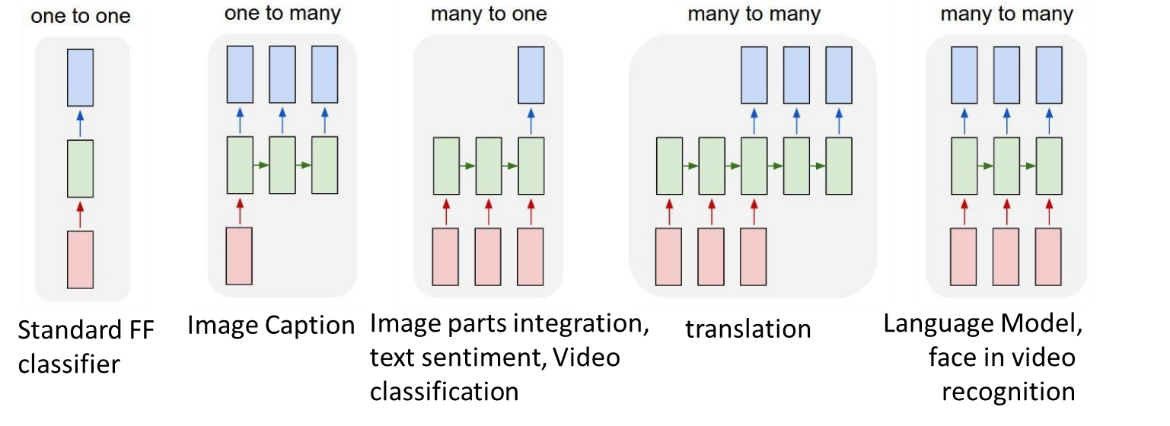
g (): פונקצית אקטיבציה לא לינארית (למשל SoftMax) שמוציאה את הפלט y.

****

y

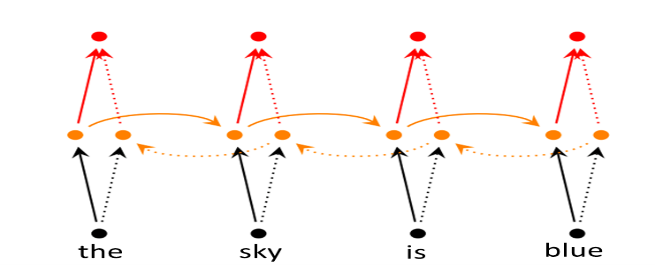


**סוגים של RNN**

****

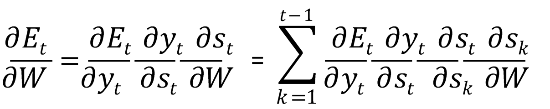
**Bi-Directional RNN**

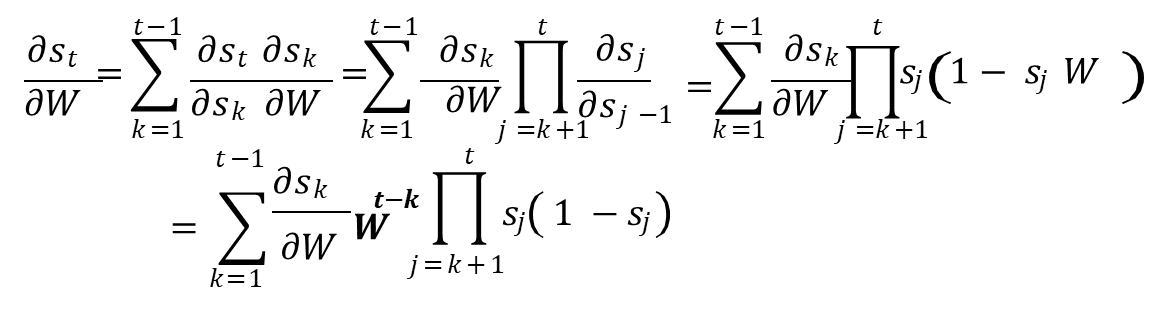
הפלט של ה RNN לא תלוי רק בקלטים שהיו עד לזמן t אלא תלוי גם בקלטים עתידיים.נשפר את השגיאה: אם הקלטים יוזרמו גם מההתחלה לסוף וגם מהסוף להתחלה.

****דוגמא לשימוש: ניחוש המילה החסרה – כדאי לתת כקלט המילים שהופיעו לפני המילה החסרה (בסדר הזמן)  
ואת המילים שאחריה בסדר הפוך.

ארכיטקטורה הבנויה מ2 RNN המקבלים את אותו קלט בסדר הפוך. שכבת הפלט מחושבת על פי שכבות הנסתרות של שתיהם

**בעיית הגראדיינטים הנעלמים והמתפוצצים ב-RNN**

**נוסחה לשיפור את המשקולות ב-RNN:**

****

נשים לב ל- 𝑊𝑡−𝑘 ,כאשר התלות בזמן תיהיה גדולה האקספוננת יגדל:

אם הגרדיאנט יעלם,ישאף לאפס.

אם הגרדיאנט "יתפוצץ",השיפור רק יעלה את גודל המשקולת.

הסבר,בכל קליטה של הקלט אנו מתבססים על הקלט הקודם,לכן ככל שיהיו יותר קלטים התלות תגדל ,כי ה-t גדל. השפעות קרובות בזמן, משפיעות, השפעות רחוקות בזמן נעלמות (או מתפוצצות).

**פתרון לבעיית הגראדיאנטים: Truncated BP**

מגבילים את כמות ה Foldout, למשל עד =k35.כלומר נגביל את סדרת הזמן. גם אם סידרת הזמן גדולה מ-35, מחלחלים אחורה שגיאות בצורה מוגבלת בד"כ כל k קלטים, זו סידרת קלטים שעושים לה unfolding לרשת FF בעומק k שכבות  
ומבצעים חילחול שגיאה BP ב-k השכבות בלבד.

Unfolding ברשת RNN –זו שיטה שפורשת אותה לרשת של FF בשבילי הנוחות,הצגה קומפקטית של גרף. פעולה זו אינה מוסיפה משקולות לרשת!

**כל k קלטים, מבצעים את BP ב-k השכבות :**

* אוספים batch של m סדרות של k קלטים. האיסוף הוא של m סדרות באורך k,הסדרות הם ממיקומים שונים בקבוצת האימון.
* מבצעים unfolding לרשת FF של k הקלטים (k שכבות) במיני batch.
* מבצעים BP ועידכון משקולות לכל ה batch ביחד
* ב batch הבא יבחרו m סדרות חדשות של k קלטים
* כל batch הוא זוגות (קלט + target ).

**LSTM - Long Short Term Memory**

יחידת ה-LSTM הוא מעין נוירון נסתר ברשת RNN.מחושב בצורה מיוחדת כפונקציה של הקלט x והמצב הקודם :

רכיב LSTM מכיל בתוכו 4 נוירונים נסתרים ומייצר 2 פלטים סקלרים:

* 1. מצב :(Cell-State) נושא איתו זיכרון ארוך טווח
  2. פלט : מספר (-1,1) המחושב על פי המצב האחרון של

שכבה של n רכיבי LSTM שקולה ל 4n נוירונים נסתרים המתקשרים זה לזה באופן חכם כדי ליצר פלט (ב-2 וואריאציות) : המצב חדש c ואותו הפלט h דחוס ב-(-1,1)

* הפעולות המוקפות בעיגול: + x tanh נעשות element-wise
* הכניסה ל-4 הנוירונים הנסתרים היא בעזרת מכפלת וקטור משקולות בוקטור שמתקבל משרשור ו-

**שער השכחה**

שער השיכחה: נוירון סיגמויד שקובע כמה נשמור מתוך המצב הקודם

הקלט שלו זה הקלט החדש וגם המצב הקודם במצב מנורמל .(-1,1)

**שער הקלט**

המצב החדש הוא תוצאה של tanh על הקלט x והמצב הקודם המנורמל אולם לא חושפים אותו ישירות החוצה:

**:** **שער הקלט (שער הוספה)-** מחשב איזה חלק מהמצב החדש ילקח ויתווסף למה שנשאר מהמצב הקודם.

**המצב החדש**

**: המצב החדש:** המצב הישן שנשכח בחלקו + חלק מהמצב החדש.

**LSTM יכול לחשב מצב חדש כמו RNN פשוטה ע"י שער שיכחה והוספה אחת.**

**שער הפלט**

החדש עובר נירמול (squashing) בעזרת element-wise tanh,אך רק חלק ממנו נחשף כ-h

שער הפלט : קובע כיצד "ישכח" המנורמל בהתבסס על נוירון סיגמויד.

**נחבר את כל השערים:**

רכיב ה LSTM מיצר מצב חדש C ופלט h:

המצב החדש נוצר ע"י מישקול המצב הנוכחי עם המועמד למצב חדש. המצב הנוכחי ממושקל בעזרת שער שיכחה   
והמועמד למצב חדש ממושקל באמצעות שער ההוספה.

בסוף משתמש הרכיב בשער הפלט כדי לחשב את הפלט בהסתמך על המצב החדש.

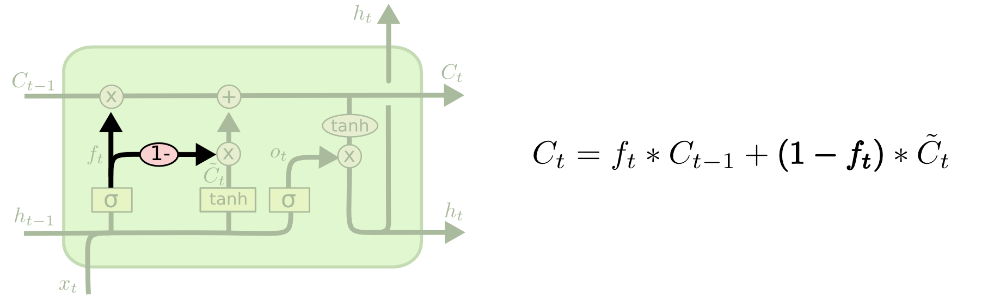
3 השערים הם נוירונים לוגיסטים והמועמד למצב חדש הוא נוירון tanh.

הנוירונים מושפעים מהקלט החדש והמצב הקודם, משקולותיהם נלמדים.

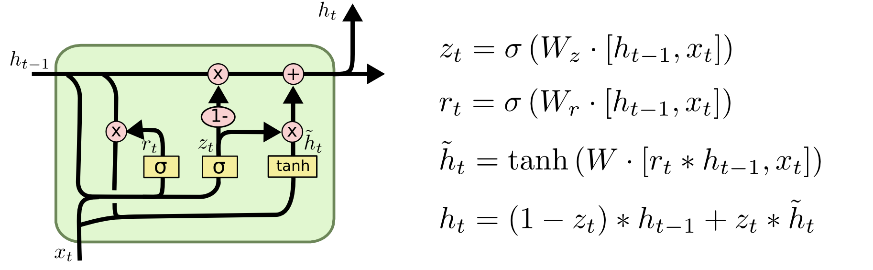
**Variants of LTM: PeepHoles Connections**

לשערים (Gates)מותר להסתכל על המצב הלא מצונזר .(c)

**Variants of LTM: Coupled Forget and Input Gate**

חוסכים שער אחד: משתמשים בשער ה-forget אבל השער להוספה (קלט) מחושב ע"י .

**Variants of LTM: Gated Recurrent Network (GRU)**

שינויים יותר רדיקלים, חיסכון בשערים, באקטיבציות ובמשקולות:

* רק פלט אחד ממוזג : המצב החדש גם

חוזר כמשוב וגם הולך לפלט.

* שער אחד z עבור שיכחה והוספה .(1-z)
* שער שיכחה נפרד עבור המצב הקודם

כשהוא נכנס לנוירון ה-tanh.

במחקרים אמפיריים מסוימים נראה כי GRU משיג תוצאות דומות ל-LSTM (שגיאת הכללה) ,אך יעיל יותר מבחינה חישובית.בינתיים אין מסקנה גורפת ו-LSTM עדיין פופולרי למרות ש-GRU חדש, ופשוט יותר לחישוב.

