שיעור 4

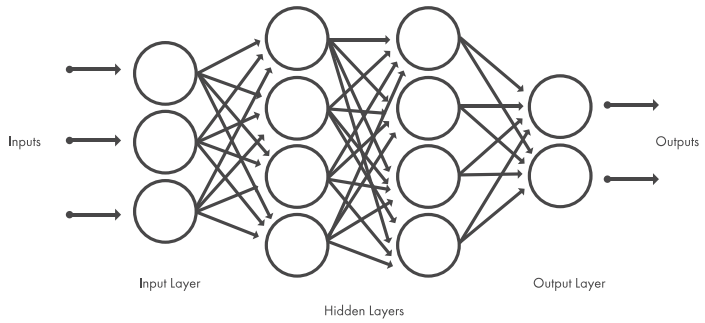
30 באפריל 2019

1 רשתות נוירונים

בשיעור זה נתחיל את הלימוד שלנו על רשתות נוירונים. רשתות נוירונים הן כר פורה למחקר גם היום, ומקיימות תכונות מעניינות רבות. כאלגוריתם למידה זהו ככל הנראה האלגוריתם הנפוץ, החזק והגמיש ביותר. הן משמשות היום כמעט בכל תחום: החל במשימות תרגום וניתוח טקסט, דרך משימות של עיבוד תמונה ווידיאו וכלה בשימוש באלגוריתמי unsupervised ומציאת אנומליות.
באופן מפתיע משהו, יש לנו כעת את כל הכלים כדי להבין את אופן הפעולה של רשתות ולבצע ניתוחים בסיסיים עליהן. אנו צפויים להיתקל במספר סוגיות שפגשנו במידה פחותה יותר בעבר - למשל, בסוגיית התאמת היתר (over fit), בהתייחסות להתפלגות של המידע, ועוד.
בשיעור זה נעסוק ברשת הנוירונים הבסיסית ביותר - רשת fully connected עם אקטיבציה לא ליניארית. נלמד על האופן בו רשתות נוירונים מבצעות פרדיקציה וכיצד הן מתאמנות.

1.1 מבנה רשת נוירונים ותהליך החיזוי

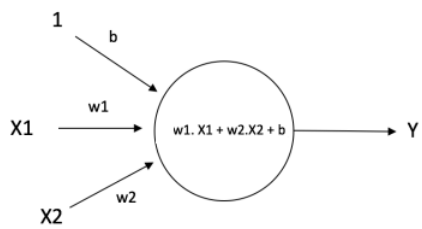
ראשית, רשת נוירונים היא מודל ממשפחת מודלי ה־supervised learning: ככזו, היא מאפשרת לנו משפחת פונקציות, אותה אנו יכולים להתאים באמצעות כיוונון הפרמטרים שלהן: ממש כפי שעשינו ברגרסיה ליניארית. כדי לאמן אותה נצטרך לקבל מידע מתויג, כאשר נרצה לאמן את המודל לחזות את התיוג.



איור 1:

רשת נוירונים מורכבת משכבות של נוירונים. נוירון הוא פונקציה, המקבלת אוסף קלטים ומוציאה אוסף פלטים. בכתיב של רשתות אנו מייצגים רשת נוירונים כגרף מכוון, כאשר כל קודקוד הוא נוירון, והקשתות מייצגות את מעבר הקלטים והפלטים בין נוירונים. לפעמים יהיה לנו נוח לדבר על המעבר של נתונים בין שני נוירונים, אך נרצה לדבר גם על מעבר המידע בין שכבות: נוכל לחשוב גם על כל שכבה כעל פונקציה, המקבלת אוסף קלטים (אוסף כל הקלטים שהנוירונים שלה מקבלים) ומחזירה אוסף פלטים (אוסף כל הפלטים שהנוירונים שלה מוציאים).
ברשת נוירונים קלט נכנס לרשת, ועובר טרנספורמציות במעבר שלו דרך הרשת. לכל נוירון ברשת ישנן הקשתות הנכנסות אליו: הן מבטאות את האופן בו הוא מקבל קלט. עבור הקלטים $x_1, ..., x_n$ שנוירון מקבל, הוא פועל עליהם כך:

$$h(x_1, ..., x_n) = \sum_{i=1}^{n} w_i \cdot x_i + b$$



איור 2:

w_i הן המשקולות אשר "מתווכות" לנו את הקלטים השונים. כל קלט שנכנס אל הנוירון נכפול במשקולת w המתאימה. הפרמטרים w_i הם פרמטרים נלמדים ברשת - בהמשך נראה כיצד אנו מתאימים אותם. גם הרכיב החופשי, ה־bias, הוא פרמטר נלמד ברשת.
נוכל לכתוב את פעולת שכבה של נוירונים על הקלטים שלה באופן וקטורי: אנו רואים כי h היא טרנספורמציה ליניארית של הקלט, יחד עם רכיב חופשי. לכן נוכל לכתוב את המעבר של וקטור דרך שכבה של רשת נוירונים כך:

$$\overrightarrow{x} \mapsto W \cdot \overrightarrow{x} + \overrightarrow{b} = \begin{pmatrix} h_1(x_1, ..., x_n) \\ \vdots \\ h_m(x_1, ..., x_n) \end{pmatrix} = \begin{pmatrix} w_{11}x_1 + ... + w_{1n}x_n + b_1 \\ \vdots \\ w_{m1}x_1 + ... + w_{mn}x_n + b_m \end{pmatrix}$$

כל רכיב בוקטור הפלט הוא ביטוי מהצורה $h(x_1, ..., x_n)$ כפי שכתבנו קודם, ומתאר את פעולת נוירון יחיד על הקלטים שהוא מקבל. בסך הכל נקבל פעולה של שכבת נוירונים שלמה על השכבה הקודמת.
בנוסף לקיבוץ הקלטים, כל נוירון מפעיל על $h(x_1, ..., x_n)$ פונקציית אקטיבציה - זו פונקציה לרוב לא ליניארית, נסמנה ב־ϕ. לכן הפלט של כל נוירון (ובפרט, אחד מהקלטים של הנוירון הבא) הוא

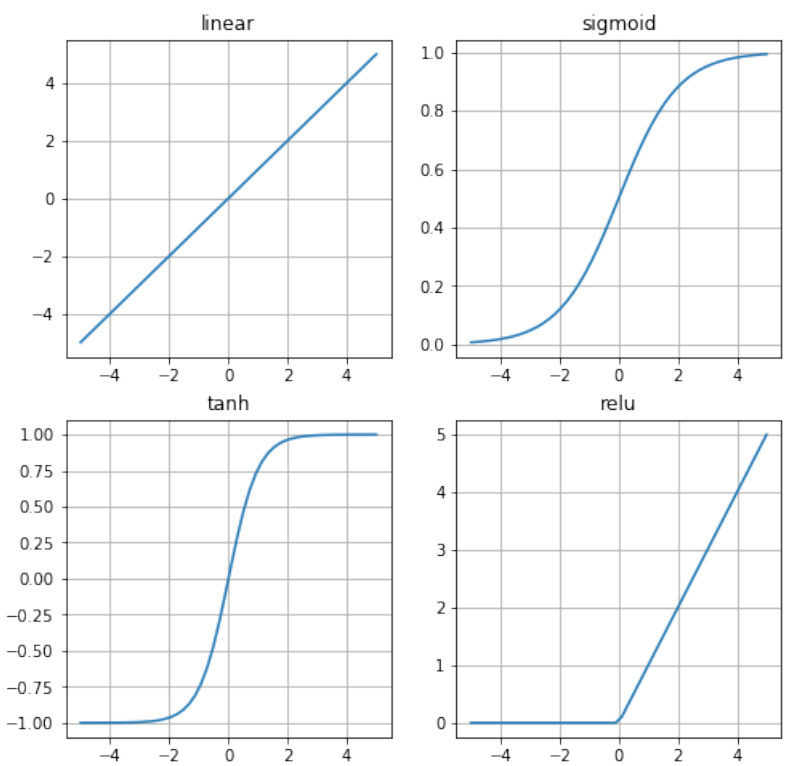
$$\phi(h(x_1, ..., x_n))$$

בכתיב הוקטורי נכתוב

$$\overrightarrow{x} \mapsto \phi(W \cdot \overrightarrow{x} + \overrightarrow{b})$$

כאשר ϕ פועלת על הוקטור המתקבל איבר־איבר.
הפונקציה ϕ ניתנת לבחירה. כמה פונקציות אקטיבציה נפוצות הן הבאות:

$$Id(x) = x$$
$$\sigma(x) = \frac{1}{1+e^x}$$
$$tanh(x)$$
$$relu(x) = max(0, x)$$



איור 3:

עבור רשת נוירונים קווית פשוטה בת שלושה נוירונים, הפלט שלה נתון לכן ע"י

$$\phi(h_2(\phi(h_1(x))))$$

מה יקרה במקרה של אקטיבציה ליניארית? נקבל כי הפלט הוא $h_2 \circ h_1(x)$, ובאופן כללי לרשתות גדולות יותר,

$$W_n \cdot (W_{n-1} \cdot (... \cdot (W_1 \overrightarrow{x} + \overrightarrow{b_1}) + ...) + \overrightarrow{b}_{n-1}) + \overrightarrow{b}_n = \Pi_{i=n}^{1} W_i \cdot \overrightarrow{x} + \overrightarrow{b}$$

כלומר, רשתות נוירונים עם אקטיבציה ליניארית ובעומק שרירותי הן בסך הכל טרנספורמציה ליניארית של הקלט עם bias, מכך שהרכבה של מטריצות היא טרנספורמציה ליניארית. זו מסקנה מעניינת - מסתבר כי לפונקצית האקטיבציה יש תפקיד חשוב בפעולת הרשת, וללא פונקציית אקטיבציה, רשת נוירונים היא למעשה מודל פשוט למדי, שאינו טוב יותר מרגרסיה ליניארית. אנו למדים מזה גם כי משפחת המודלים של רגרסיות ליניאריות בפרט מוכלת בזו של רשתות הנוירונים. למעשה, קל (לבוגרי הקורס באינפי 3) יהיה להראות כי במודל של רשת נוירונים עם אקטיבציה לא ליניארית (למשל σ) ניתן לקרב כל פונקציה חלקה. זו אחת הסיבות שרשת נוירונים הן משפחת מודלים גנרית וגמישה כל כך.
החוזק של רשתות הוא, אם־כן, באקטיבציה לא ליניארית. שימו לב כי ניתן בעיצובים מסוימים של רשתות (ארכיטקטורות) לבחור שכבת אקטיבציה שונה לכל שכבה בנפרד. ניתן לעשות זאת גם לכל נוירון בנפרד, אך הדבר אינו נהוג.
איך נדע אילו פונקציות אקטיבציה הן "טובות"? מה הן התכונות שהיינו רוצים שפונקציית אקטיבציה טובה תקיים? נוכל לענות על השאלה הזו בהמשך, באמצעות ההבנה של האופן בו רשתות נוירונים מתאמנות.
כעת אנו יודעים כיצד מתבצעת פרדיקציה ברשת: בהינתן רשת מאומנת (כלומר, עם W_i, b_i שנקבעו באופן חכם), פרדיקציה עבור וקטור הפיצ'רים x תתבצע ע"י

$$h_i(x) = W_i x + b_i$$
$$network(x) = \phi(h_n(\phi(h_{n-1}(...\phi(h_1(x))...))))$$

תהליך המעבר של וקטור דרך הרשת נקרא feed forward.

1.1.1 דוגמא - רגרסיה לוגיסטית

כזכור, המודל של רגרסיה לוגיסטית נתון לנו ע"י

$$f(x) = \sigma(a \cdot x + b)$$

כאשר σ היא פונקציית הסיגמואיד. לכן קל לראות כי גם את הרגרסיה הלוגיסטית ניתן לקבל באמצעות רשת נוירונים עם שכבת קלט ושכבת פלט בלבד, כאשר האקטיבציה של הרשת היא פונקציית הסיגמואיד.

1.2 אימון רשתות ו־back propagation

אימון רשת, ברמה בסיסית, דומה לכל אימון של מודל בו משתמשים בשיטות אופטימיזציה כדוגמאת gradient decsent: אנו מחשבים את ה־loss עבור סט הדוגמאות, גוזרים אותו לפי המשקולות שלנו ומעדכנים אותן בהתאם, באמצעות שינוי שלהן נגד כיוון הגרדיאנט. נזכיר:

$$w \mapsto w - \varepsilon \cdot \frac{\partial L}{\partial w}$$

כך גם ברשת נוירונים: אנו מעבירים את אוסף הקלטים x_α דרך הרשת ומקבלים אוסף פלטים $\hat{y}_\alpha$, כאשר אנו יודעים את אוסף ה־target, y_α. ראינו בעבר (כשעסקנו במודלים פשוטים, של רגרסיה ליניארית ורגרסיה לוגיסטית) כי מודלים מסוימים מסוגלים לתת פלט בצורת וקטור. באופן כללי, רשת נוירונים גם היא נותנת פלט וקטורי.
נתעכב על הפרטים: בהינתן דוגמא $\overrightarrow{x}$, כאשר נסמן את הפונקציה של רשת הנוירונים ב־f,

אנו יכולים לחשב את $f(\overrightarrow{x})$, ואז נוכל לחשב את ה־loss של הרשת: אם ה־Loss הוא פונקציה

$$L(s,t)$$

אזי נחשב את $L(f(\overrightarrow{x}),y)$. זהו ה־feed forward. כעת נוכל לחשוב על f כעל פונקציה של המשקולות W,b, ונוכל לגזור את הביטוי $L(f(\overrightarrow{x};W,b),y)$ ביחס ל־W,b. נקבל כי

$$\frac{\partial L}{\partial w}=\frac{\partial L}{\partial s}|_{f(\overrightarrow{x};W,b),y}\cdot\frac{\partial f(\overrightarrow{x};W,b)}{\partial w}|_{W,b}$$

השתמשנו בכלל השרשרת כדי לגזור את L ביחס למשקולת w. לתהליך זה, בו אנו פותחים את כלל השרשרת כדי לעדכן כל משקולת, אנו קוראים back propagation.

1.2.1 דוגמא ל־back propagation ברשת קווית עם אקטיבציה לא ליניארית

נתבונן ברשת עם שכבת קלט, שכבת hidden ושכבת פלט, כולן עם נוירון יחיד. נשתמש באקטיבצית סיגמואיד. נזכר כי כשהצגנו לראשונה את פונקציית הסיגמואיד למדנו על תכונה נחמדה מאוד שלה:

$$\sigma^{'}(z)=\sigma(z)\cdot(1-\sigma(z))$$

תכונה זו תקל עלינו מאוד לחשב את הנגזרות בתהליך ה־back propagation.
נכתוב את הפונקציה של הרשת, f, עם כל הפרמטרים שלה:

$$f=\phi\circ h_2\circ\phi\circ h_1$$
$$f(x)=\phi(h_2(\phi(h_1(x;w_1,b_1));w_2,b_2))=\sigma(w_2\cdot\sigma(w_1\cdot x+b_1)+b_2)$$

נניח כי ה־loss של הרשת הוא mse. נכתוב

$$\begin{aligned}L(s,t)&=(s-t)^2\\ \phi(z)&=\sigma(z)\\ h_2(r)&=w_2\cdot r+b_2\\ h_1(m)&=w_1\cdot m+b_1\end{aligned}$$

שימו לב: אני משתמש בשמות משתנים שונים להגדרה הכללית של הפונקציות בכוונה, כך שיהיה ברור מי הם המשתנים לפיהם אנו גוזרים ואת מי אנו מציבים בביטוי הנגזרת. לכן עבור הזוג (x,y) נקבל את ה־loss

$$L(f(x),y)=(f(x)-y)^2$$

נגזור את ה־loss ביחס למשקולות השונות. נתחיל מ־w_2.

$$\frac{\partial L}{\partial w_2}(f(x),y)=\frac{\partial L}{\partial s}|_{(f(x),y)}\cdot\frac{\partial f}{\partial w_2}|_x=\frac{\partial L}{\partial s}|_{(f(x),y)}\cdot\frac{\partial\sigma}{\partial z}|_{h_2(\sigma(h_1(x))}\cdot\frac{\partial h_2}{\partial w_2}|_{\sigma(h_1(x)}$$

כעת אנו יודעים כי $\frac{\partial\sigma}{\partial z}=\sigma(z)\cdot(1-\sigma(z))$, כי $\frac{\partial L}{\partial s}=2\cdot(s-t)$ וכי $\frac{\partial h_2}{\partial w_2}=r$. לכן,

$$\frac{\partial L}{\partial w_2}(f(x),y)=\underbrace{2\cdot(f(x)-y)}_{2\cdot(s-t)}\cdot\underbrace{\sigma(h_2(\sigma(h_1(x)))\cdot(1-\sigma(h_2(\sigma(h_1(x))))}_{\sigma(z)\cdot(1-\sigma(z))}\cdot\underbrace{\sigma(h_1(x)}_{r}$$

נוכל באותו האופן לקבל את הנגזרת לפי כל פרמטר אחר:

$$\frac{\partial L}{\partial b_2}(f(x),y) = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial f}{\partial b_2}|_x = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial \sigma}{\partial z}|_{h_2(\sigma(h_1(x))} \cdot \frac{\partial h_2}{\partial b_2}|_{\sigma(h_1(x)}$$
$$\frac{\partial L}{\partial w_1}(f(x),y) = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial f}{\partial w_1}|_x = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial \sigma}{\partial z}|_{h_2(\sigma(h_1(x))} \cdot \frac{\partial h_2}{\partial r}|_{\sigma(h_1(x)} \cdot \frac{\partial \sigma}{\partial z}|_{h_1(x)} \cdot \frac{\partial h_1}{\partial w_1}|_x$$
$$\frac{\partial L}{\partial b_1}(f(x),y) = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial f}{\partial b_1}|_x = \frac{\partial L}{\partial s}|_{(f(x),y)} \cdot \frac{\partial \sigma}{\partial z}|_{h_2(\sigma(h_1(x))} \cdot \frac{\partial h_2}{\partial r}|_{\sigma(h_1(x)} \cdot \frac{\partial \sigma}{\partial z}|_{h_1(x)} \cdot \frac{\partial h_1}{\partial b_1}|_x$$

חישבנו את כל הנגזרות של הרשת בפרמטרים השונים שלה. כעת נוכל לעדכן כל פרמטר ב־gradient decsent ע"י

$$w_2 \mapsto w_2 - \varepsilon \cdot \frac{\partial L}{\partial w_2}$$
$$b_2 \mapsto b_2 - \varepsilon \cdot \frac{\partial L}{\partial b_2}$$
$$w_1 \mapsto w_1 - \varepsilon \cdot \frac{\partial L}{\partial w_1}$$
$$b_1 \mapsto b_1 - \varepsilon \cdot \frac{\partial L}{\partial b_1}$$

בכך נשלים צעד אימון אחד. "חיפפנו" רק בחלק אחד: כזכור, ה־Loss שלנו הוא לא על דוגמא יחידה, אלא על כל הדוגמאות. לשם כך עלינו להחליף את L בביטוי מהצורה

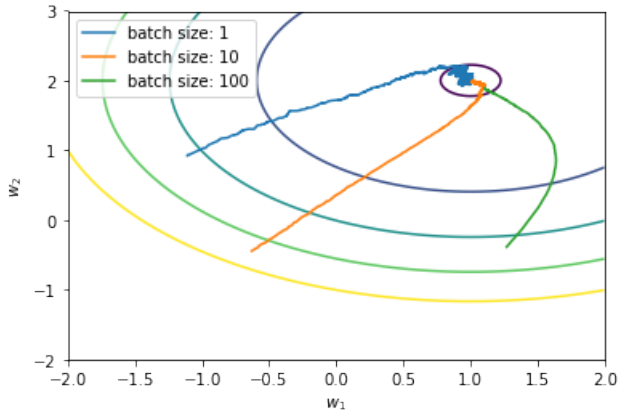
$$\sum_i (f(x_i) - y_i)^2$$

ולגזור באותו האופן.
נשים לב כי אמנם הגזירה היא חוויה מפוקפקת, אך כשמאמנים רשת יש לחשב את הנגזרות פעם אחת בלבד: כאשר "מקמפלים" אותה, עוד לפני תחילת האימון. במהלך האימון עצמו הרשת יכולה להציב את הערכים המתקבלים מ־x ב־feed forward בתוך הביטויים של הנגזרות לחישוב מהיר.
בהמשך נראה ארכיטקטורות רשת מסובכות יותר ויותר. לא נדגים בכל פעם מחדש את תהליך ה־back propagation, אך העיקרון בכל הדוגמאות הללו יהיה זהה: יש לכתוב את הפונקציה ולגזור אותה בזהירות בכלל השרשרת. בפרט זהו התהליך שנרצה לעשות לרשתות fully connected כלליות בהן עסקנו עד כה.
נסיק מסקנה פשוטה מההליך כולו: אם בעבר למדנו ש־"machine learning" הוא שם מפואר לפתרון בעיות מינימום, כעת אנו למדים ש־"back propagation" הוא שם מפואר לכלל השרשרת.

1.3 Stochastic Gradient Decsent

כשהגדרנו gradient descent הגדרנו אותו על כל המידע בסט האימון. עולה מכך בעיה משמעותית, שמקבלת משנה תוקף בכל העיסוק ברשתות: סט האימון עשוי להיות גדול מאוד, ומעבר על כולו ־ איטי מאוד. נרצה לקבל "פידבק" לתוך הרשת לעתים תכופות, ללא צורך במעבר על כל סט האימון. לשם כך ניתן לחשב את ה־loss, ובהתאם גם את הגרדיאנטים (הנגזרות ברשת), על סמך מדגם קטן של המידע ־ batch. נחלק את סט האימון שלנו ל־batch־ים, ונתאמן באמצעות כל batch בנפרד.
ברמה השטחית, אימון באמצעות batch הוא מדויק פחות, כיוון שכל batch מתיימר לייצג את ההתפלגות האמיתית של המידע, אך בהכרח אינו עושה זאת. אולם הוא מאפשר אימון מהיר יותר (ובמובנים מסוימים גמיש יותר). לכן גודל ה־ batch בו נשתמש הוא בעל חשיבות לקביעת אופי האימון. ב־batch־ים קטנים לדוגמאות בודדות יש יותר חשיבות, דבר שעשוי להקטין את הדיוק בלמידה, אך נקבל גרדיאנטים לתוך הרשת לעתים קרובות יותר.
לאימון ב־gradient decsent עם batch־ים נקרא Stochastic Gradient Decsent, כיוון

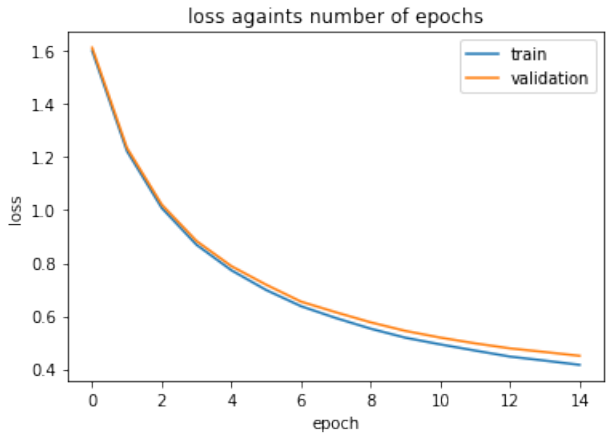
שיש בו רכיב אקראי.



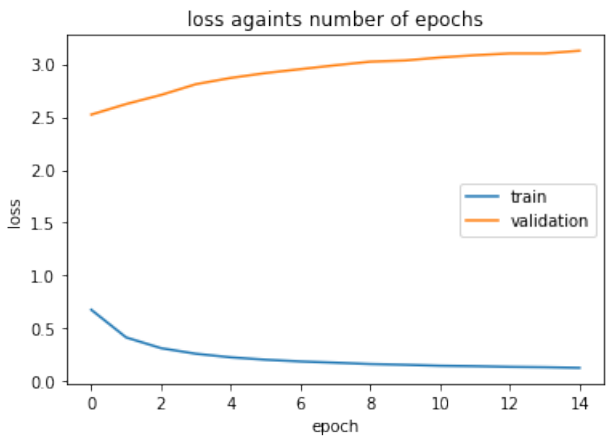
איור 4:

1.4 ניתוח האימון של רשתות ו־overfit

כפי שציינו, רשת נוירונים היא אלגוריתם חזק, במובן זה שהיא יכולה להתאים פונקציות רבות. משום כך בשימוש בו אנו גם חשופים להתאמת יתר. עם זאת, אימון של רשתות, בשונה מאימון של אלגוריתמים לומדים אחרים, אינו תהליך סופי וברור, אלא עדכון משקולות חוזר ונשנה באמצעות gradient decsent. משום כך נוכל לשלוט באימון בצורה טובה יותר, ולהפסיקו באם אנו מזהים שאנו מגיעים להתאמת יתר. באופן כללי באימון של רשתות נרצה להתבונן בגרף של ה־loss כתלות בפרמטר זמן כלשהו (למשל, כמות ה־batch־ים שהתאמנו עליהם), עבור סט האימון ועבור סט ואלידציה. מהתבוננות זו נוכל להבין, מחד, מתי האימון ממצה את עצמו, כלומר מתי ה־loss מפסיק להשתפר, וגם נוכל לדעת מתי אנו נקלעים ל־over fit ־ כאשר ה־loss על סט האימון יורד, אך לא כך ה־loss על סט הואלידציה.



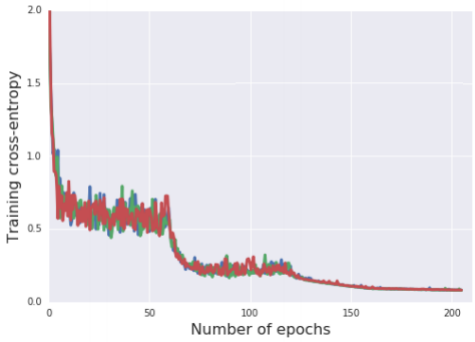
איור 5:



איור 6:

פעמים רבות נרצה לצייר במהלך האימון גם את הערך של מטריקות (מדדים) נוספות שנרצה למדוד את עצמנו עליהן. לפעמים המדדים שנתעניין בהם הם למעשה הגודל "האמיתי" שנרצה למקסם, אולם לא נוכל להשתמש בו בתור loss - מצב זה קורה לרוב כיוון שהמדד שלנו אינו גזיר כתלות במשקולות. למשל, accuracy, כמה פעמים צדקנו מתוך כלל התחזיות שלנו, הוא מדד שכזה.

נקודה אחרונה: כזכור לכם, בשימוש ב־gradient decsent אנו מגדירים learning rate. ראינו גם כי learning rate גבוה מדי עשוי למנוע מאיתנו להתכנס, בעוד ב־learning rate נמוך מדי ההתכנסות תהיה איטית מאוד. נוכל להשתמש ביכולת שלנו לנטר את ה־loss תוך כדי האימון כדי להבין האם הגענו להתכנסות מקומית, ומוטב שנקטין את ה־learning rate.



איור 7:

1.5 אפיון של פונקציות אקטיבציה

אנו מבינים כעת כיצד פועלת רשת וכיצד היא מתאמנת. ראינו גם כיצד נראה תהליך האימון עצמו. כזכור, ראינו כמה פונקציות אקטיבציה, אך טרם ענינו על השאלה - מה מאפיין פונקציית אקטיבציה טובה? אילו תכונות על פונקציית אקטיבציה לקיים?
ראינו כי פונקציית אקטיבציה ליניארית תוביל לכך שהרשת שלנו תוכל ללמוד טרנספורמציות ליניאריות בלבד. לכן באופן כללי, פונקציית אקטיבציה טובה היא פונקציית אקטיבציה שאינה ליניארית.
ראינו גם פרמטר "חישובי" שעשוי לעניין אותנו - למשל, הנגזרת של פונקציית הסיגמואיד בנקודה קלה מאוד לחישוב, בהינתן ערך הפונקציה בנקודה. תכונות שכאלו חשובות לאימון רשת "בחיים האמיתיים", תחת מגבלה של כוח מחשוב.
תכונות נוספות של פונקציית האקטיבציה נרצה לגזור מתוך האופן בו אנו מאמנים את הרשת. בפרט, על פונקציית האקטיבציה להיות גזירה (כמעט בכל מקום, לכל הפחות. מדוע דרישה זו מספיקה?).
נזכר כי בביטוי שקיבלנו מכלל השרשרת מופיעות מכפלות של נגזרות של פונקציית האקטיבציה. לכן הגיוני לדרוש חסימות כלשהי על הנגזרת, כך שלא תתבדר ל־∞.
לבסוף, כדי לפשט את תהליך התאמת הפרמטרים נהוג לדרוש פונקציית אקטיבציה מונוטונית - כדי למנוע מצב של קפיצה בין נקודות שונות של הפונקציה שיתנו את אותו הערך בטווח (חישבו, למשל, מה עשוי לקרות עבור פונקציית האקטיבציה $f(x) = x^2$ ברשת הפשוטה ביותר - נוירון כניסה ונוירון יציאה יחידים, אם ה־learning rate גדול מדי).
למעשה, שאלת פונקציות האקטיבציה ברשת היא שאלה מעניינת מאוד ומסובכת מאוד. חוקרים בימינו מתמודדים עם השאלה, כיצד נראה משטח השגיאה של רשת נוירונים, בהינתן פונקציית אקטיבציה מסוימת? משטח השגיאה הוא פונקציית ה־loss כתלות במשקולות של הרשת - הבנה טובה יותר שלו תוביל להבנה טובה יותר של שיטות האימון של רשתות. גם השאלה, כיצד נראה משטח השגיאה של רשת עם אקטיבציות ליניאריות, אינה פשוטה במיוחד.

1.6 Optimizers

עד כה הצגנו שיטה יחידה לביצוע אופטימיזציה (optimizer) - gradient decsent. למעשה ישנן שיטות רבות לבצע אופטימיזציה (וודאי למדתם עליהן באופן כלשהו בקורס של ארזי), אף שלמרבה האכזבה, הנפוצות ביניהן הן וריאציות קלות על gradient decsent. למשל,

שימוש במומנטום:
נחשוב על הגרדיאנט לא כעל "שינוי במקום", אלא כעל שינוי במהירות. במילים אחרות, כאשר אנו מחשבים את "כיוון ההתקדמות", אנו מוסיפים אותו, עם משקל כלשהו, לכיוון הקודם, ומתקדמים בכיוון החדש. אם הנוסחה המוכרת שלנו נראיתה כך:

$$w \mapsto w - \varepsilon \cdot \nabla_w L$$

אזי הנוסחה החדשה נראית כך:

$$\begin{gathered} v = \alpha \cdot v - \varepsilon \cdot \nabla_w L \\ w \mapsto w + v \end{gathered}$$

שיטה זו מאופיינת ביציבות רבה יותר של האימון, והיא רגישה פחות לדוגמאות חריגות. שימו לב שכעת יש לקבוע שני פרמטרים השולטים באופי האימון - ה־learning rate והקבוע α, הקובע לנו את "השפעת העבר" על כיוון ההתקדמות החדש. ככל ש־α גדול יותר כך קשה לנו יותר לשנות את כיוון ההתקדמות.
ניתן לשכלל את gradient decsent בצורות רבות. למשל, ניתן לשנות באופן מושכל את ה־learning rate במהלך האימון. ניתן גם לקבוע learning rate שונה לכל משקולת, ועוד ועוד.

1.7 סיכום ומילה לגבי ההמשך

למדנו בשיעור זה מהי רשת נוירונים. למדנו כיצד היא פועלת ומפיקה תחזיות, כיצד מאמנים אותה, והתחלנו להבין את המורכבות שרשתות נוירונים מציבות לנו, כמו גם את הפוטנציאל הרב שיש בהן. בשיעור התמקדנו בארכיטקטורה יחידה של רשתות נוירונים - ארכיטקטורת fully connected, בה כל שכבה מחוברת לשכבה הבאה אחריה בכל הקשתות האפשריות. בשיעורים הקרובים נכיר ארכיטקטורות אחרות, שמטרתן להתמודד עם בעיות מעולמות ספציפיים יותר. בפרט, נכיר ארכיטקטורות מוכרות להתמודדות עם תמונות ולהתמודדות עם מידע סדרתי (למשל, משפטים בשפה טבעית).