

מבוא לבינה מלאכותית – תרגול 4

נושאים:

- סיווג Multiclass
 - רגרסיה לוגיסטית
 - רשתות נוירונים
-

סיווג Multiclass:

לפני שניכנס לענייני רשתות נוירונים, נלמד בקצרה איך עוברים מלמידה עם מסווג לינארי בודד וסיווג בינארי ללמידה עם הרבה מסווגים לינאריים וסיווג לאחד מבין k מחלקות (שימו לב שמה שנלמד תופס לא רק על מסווגים לינאריים. במקום כל מסווג לינארי אפשר לשים כל פונקציה לסיווג, כלומר מחליפים את $\vec{w} \cdot \vec{x} + b$ ב- $f(x)$ מתאים, ואז מקבלים את דרך ההתנהגות בסיווג מולטיקלאס למחלקת השערות כללית).
בסיווג הבינארי, התיוג שלנו למחלקות היה:

$$\hat{y} = \text{sign}(\vec{w} \cdot \vec{x} + b)$$

באופן שקול שיהיה שימושי להמשיך:

$$\hat{y} = \arg \max_{y \in \{\pm 1\}} y (\vec{w} \cdot \vec{x} + b)$$

לסיווג מולטיקלאס, יש שתי גישות עיקריות להתמודדות: One vs. One (OvO) - One vs. Rest (OvR). בגישה הראשונה (פחות יעילה ופחות נפוצה), מבצעים $\binom{k}{2}$ פעמים סיווג בינארי – לכל זוג מחלקות שונות מתוך ה- k הקיימות, מבצעים משימת סיווג בינארי על כל הדאטא. לבסוף, כאשר רוצים לסווג כל דגימה בודדת, בוחרים לפי הכרעת הרוב – התיוג החזוי שניתן הכי הרבה לדגימה הוא החיזוי הסופי.

בגישה השנייה, נשתמש ב- k וקטורים למשקולות, אחד לכל מחלקה, $\vec{w}^1, \dots, \vec{w}^k$. כל אחד מהמסווגים אחראי על סיווג של אחת מבין שתי אפשרויות – שייך למחלקה המתאימה, או שייך לשאר המחלקות. החיזוי המתאים יהיה:

$$\hat{y} = \arg \max_{i \in \{1, \dots, k\}} \vec{w}^i \cdot \vec{x} + b$$

כלומר, המסווג שהדגימה מתאימה לו הכי טוב יספק את החיזוי עבור הדגימה.

עיקר העניין שלנו בחלק הזה הוא שבמקום לבצע מכפלה סקלרית של כל וקטור לבדו ב- \vec{x} , אפשר לכתוב את אוסף וקטורי המשקולות לכל המחלקות יחד בתור מטריצה $W \in \mathbb{R}^{k \times d}$, כאשר d מימד הקלט ואת המקדמים החופשיים (biases) בתור וקטור $\vec{b} \in \mathbb{R}$, ואז עדיין

$$\hat{y} = \arg \max_{i \in \{1, \dots, k\}} (W \cdot \vec{x} + \vec{b})_i$$

הערה – האלגוריתמים שלמדנו משתנים מעט בצעד העדכון שלהם במעבר בין סיווג בינארי למולטיקלאס.

העדכון ב- perceptron, אם טעינו בסיווג, הוא:

$$\vec{w}^{y(t+1)} = \vec{w}^{y(t)} + \vec{x}$$

$$\vec{w}^{\hat{y}(t+1)} = \vec{w}^{\hat{y}(t)} - \vec{x}$$

(אם יש קצב לימוד שונה מ-1 אז במקום \vec{x} יהיה שם $\eta^{(t)} \vec{x}$)

ולגבי SVM, בעקבות שינוי בפונקציית ה- hinge loss:

$$L = \max\{0, 1 - (\vec{w}^y \cdot \vec{x} + b) + (\vec{w}^{y'} \cdot \vec{x} + b)\}$$

$$\hat{y}' = \arg \max_{\substack{i \in \{1, \dots, k\} \\ i \neq y}} \vec{w}^i \cdot \vec{x} + b$$

העדכון של SVM, אם טעינו, יהיה:

$$\vec{w}^{y(t+1)} = (1 - \lambda \eta^{(t)}) \vec{w}^{y(t)} + \eta^{(t)} \vec{x}$$

$$\vec{w}^{\hat{y}'(t+1)} = (1 - \lambda \eta^{(t)}) \vec{w}^{\hat{y}'(t)} - \eta^{(t)} \vec{x}$$

$$\vec{w}^{\tilde{y}(t+1)} = (1 - \lambda \eta^{(t)}) \vec{w}^{\tilde{y}(t)} \quad \forall \tilde{y} \neq y, \hat{y}'$$

רגרסיה לוגיסטית:

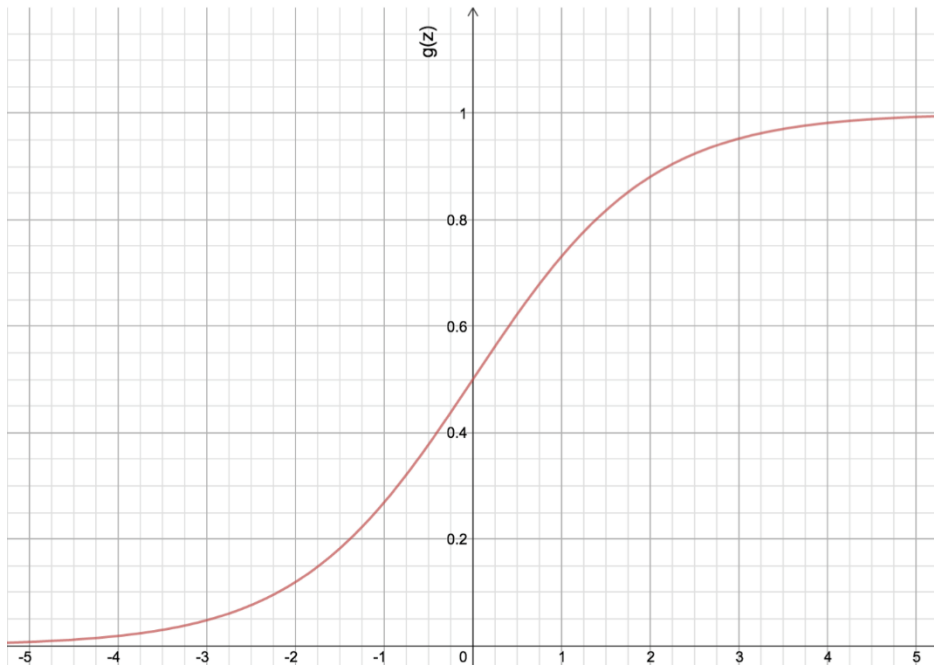
כבר בתרגיל 1 ובשיעור שעבר שמנו לב שמסווגים לינאריים הם אמנם פשוטים לשימוש ולמידה, אבל הם לא מאוד חזקים. ראינו פתרון אחד לבעיה – שימוש בפונקציית גרעין, שלמעשה מעתיקה את מרחב הקלט למרחב פשוט יותר שבו ניתן לבצע סיווג.

כעת נראה פתרון שני לבעיה:

נשים לב שאפילו הרכבה של פונקציה לא לינארית פשוטה על מסווג לינארי משפרת את היכולת לפתור בעיות: בדקו בעצמכם כי המסווג $f(x) = \max(0, x) \circ (\vec{w} \cdot \vec{x} + b)$ מצליח לפתור את בעיית ה-XOR.

ברגרסיה לוגיסטית, משתמשים בתכונה הזאת ובמקום הפונקציה הקודמת לוקחים את פונקציית הסיגמואיד:

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



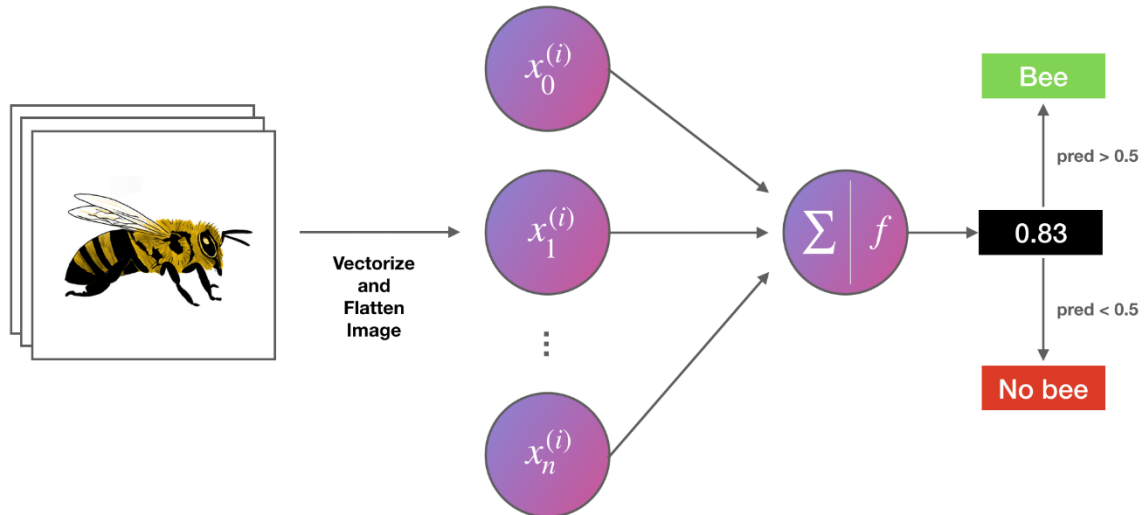
ברגרסיה לוגיסטית, מבצעים סיווג בינארי $\{0, 1\}$ באמצעות מחלקת השערות של הפונקציות:

$$h(x) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{x} + b}}$$

עם פונקציית cross-entropy loss:

$$L = - \sum_i [y_i \ln h(x_i) + (1 - y_i) \ln(1 - h(x_i))]$$

פונקציית ה- loss הזו דואגת שמה שצריך להיות מתויג 1 יהיה קרוב בחיזוי שלו ל-1 ורחוק מ-0. נשים לב שכאן לא קיבלנו חיזוי "קשיח" מפונקציית סימן או מקסימום, אלא חיזוי רציף יותר והסיבה היא כדי שיהיה אפשר לגזור את הפונקציה לפי w, b , שזה הבסיס לאלגוריתמי למידת הפרמטרים. כעת, כדי להבין טוב יותר את הנושא הבא, נייצג את המסווג בצורה הבאה, שכאן תיראה קצת מוזרה אבל בהמשך נבין רשתות נוירונים בעזרתה:



זהו בעצם המודל לרגרסיה לוגיסטית, כאשר:

- נכנס הקלט ה- i , שהוא וקטור $\vec{x}^{(i)} \in \mathbb{R}^{n+1}$ (למה $n + 1$? כי לאחד המימדים הערך הוא 1 וה- bias מוכפל בו). הקודקודים המתאימים מחזיקים את הערכים.
- על החיצים "יושבים" משקלים w_0, \dots, w_{n-1}, b (לא נראים בתמונה), כך שכשמועברים הערכים מהקודקודים $0, \dots, n$, כל אחד מהם מוכפל במשקל המתאים.
- בקודקוד שאליו נכנסים החיצים, המכפלות נסכמות ($\sum_i w_i x_i = \vec{w} \cdot \vec{x}$) ואחר כך מופעלת עליהם הפונקציה הלא-לינארית f (ברגרסיה לוגיסטית, זו פונקציית הסיגמואיד).
- התוצאה של הפעלת הפונקציה הלא לינארית משמשת לסיווג הבינארי, וכן לחישוב ה- loss שלא מופיע כאן.

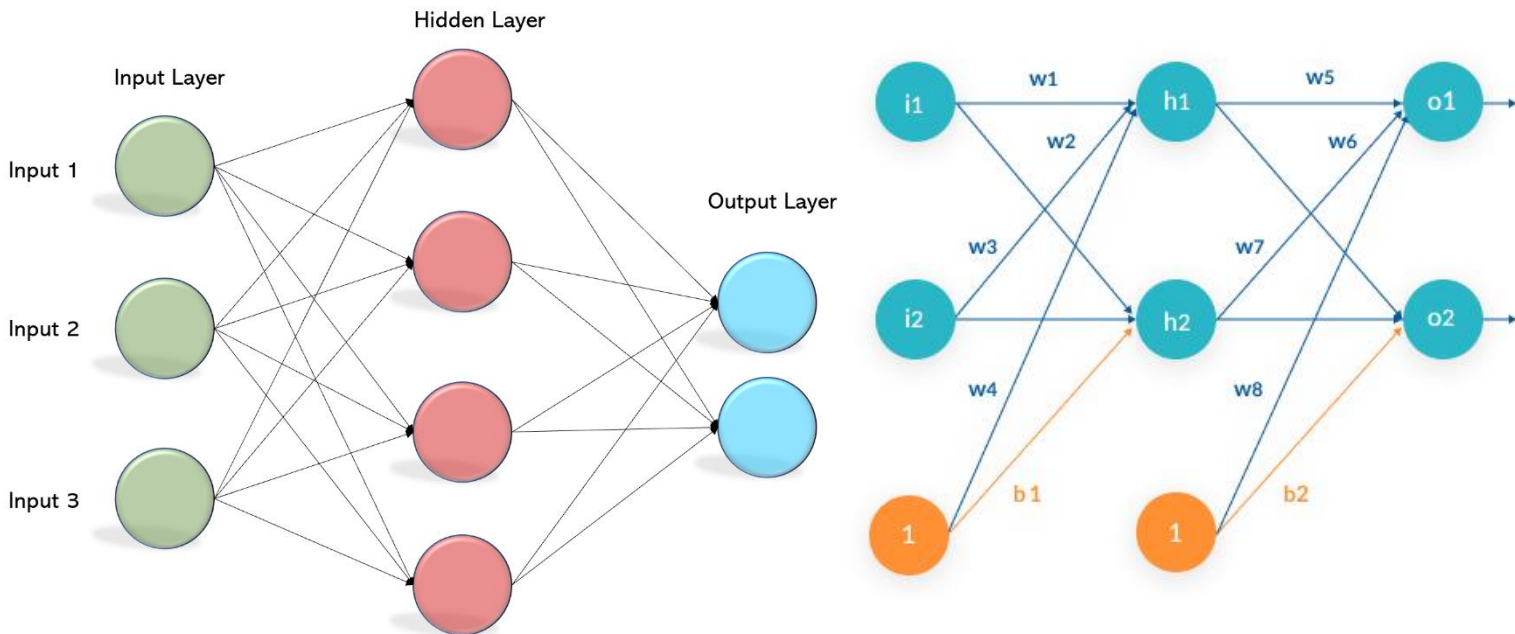
רשתות נוירונים:

את המודל הפשוט יותר שראינו קודם, אפשר לסבך יותר במטרה לקבל יכולת ייצוג רחבה יותר – רשת נוירונים.

באופן כללי, רשת נוירונים היא גרף מכון עם קשתות ממושקלות. הקודקודים של הגרף נקראים גם **נוירונים**, ובכל אחד מהם מוחזק ערך מספרי ממשי. הערכים שעל הנוירונים מועברים על פני הקשתות כך שקודקוד אליו נכנסות קשתות יקבל ערך ששווה לסכום ממושקל של ערכי הקודקודים מהן מגיעות הקשתות, כפול משקלי הקשתות.

ספציפית, כאן נדבר על המודל הפשוט ביותר לרשת נוירונים, שנקרא Feed-Forward Neural Network (FFN), וידוע גם כ- Multi-Layer Perceptrons (MLP) או Fully Connected Networks (FC).

מודל זה הוא אוסף של **שכבות של נוירונים (layers)**, כאשר מזינים לכל שכבה את הפלט מהשכבה הקודמת. השכבה ההתחלתית היא **שכבת הקלט (input layer)**, הסופית היא **שכבת הפלט (output layer)** וביניהן נמצאות **שכבות חבויות (hidden layers)**.



כאמור, לכל קודקוד n בשכבה l , מוזן מהשכבה הקודמת סכום ממושקל של הערכים מהקודקודים בשכבה הקודמת. על הערך המתקבל, מתבצעת פעולה לא ליניארית על ידי **פונקציית אקטיבציה f** :

$$z_n^{(l)} = \sum_i W_{i,n}^{(l-1)} o_i^{(l-1)}$$

$$o_n^{(l)} = f(z_n^{(l)})$$

פונקציות האקטיבציה המופעלות הן פונקציות לא לינאריות. השם שלהן נובע מתפקידן – לקבוע כמה "פעיל" כל נוירון ברשת. בזכותן, יכולת הייצוג של רשת נוירונים היא רחבה מאוד: יש משפט (לא קריטי לנו אבל סתם כדי לראות את היכולת של רשתות נוירונים) שקובע שכל פונקציה מדידה בורל ניתנת לקירוב טוב כרצוננו על ידי רשת נוירונים עם שכבה חבויה אחת (אולם, כמות הנוירונים בשכבה החבויה יכולה להיות עצומה, ולכן לוקחים במקום שכבה חבויה אחת מספר שכבות בגדלים קטנים יותר).

פונקציות אקטיבציה בולטות:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad -$$

$$ReLU(x) = \max\{0, x\} \quad -$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad -$$

יש המון פונקציות נוספות. העיקר בהן הוא שיהיו לא לינאריות וגזירות ברציפות (אולי למעט במספר נקודות סופי).

בשכבת הפלט צריכים להיות מספר נוירונים כגודל הפלט שנרצה – אם נבצע סיווג, יוכלו להיות שם מספר נוירונים כמספר המחלקות (אם כי עבור סיווג בינארי אפשר להסתפק בנוירון בודד, כמו שעשינו ברגרסיה הלוגיסטית), ואם נבצע רגרסיה, הפלט צריך להיות בגודל מתאים לגודל וקטור התיוגים (לרוב זה וקטור יחיד).

אם מדובר במשימת סיווג, אז הפלט עובר דרך פונקציה אחרונה, במטרה להפוך את וקטור הפלטים (שהוא אוסף הנוירונים בשכבת הפלט) לווקטור הסתברויות, שבאמצעותו נוכל ללמוד:

אם יש לנו נוירון יחיד, אז כמו קודם, הפעלת פונקציית סיגמואיד גורמת לפלט לייצג את הסיכוי, על סמך המודל, שהקלט מתויג 1.

אם יש לנו יותר מנוירון יחיד בשכבת הפלט, נפעיל את פונקציית ה-**softmax**:

$$softmax(\vec{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

והתיוג החזוי המתאים לקלט יהיה ה- $\arg \max$ של הפלטים היוצאים, כלומר האינדקס שמקבל את הסיכוי הגבוה ביותר להיות התיוג של הקלט על סמך המודל.

האימון של רשתות נוירונים מתבצע באמצעות שיטות מבוססות גראדיינט. כזכור, אנחנו מנסים למזער פונקציית loss, כתלות בכל המשקולות וה-biases של הרשת. הדרך לבצע את המזעור הזה נקראת **backpropagation**:

בהינתן ערך של פונקציית ה-loss, מנסים להעריך את ערכי הגראדיינטים של הפונקציה לפי כל אחד מהפרמטרים של הרשת: $\frac{\partial L}{\partial W_{i,j}^{(l)}}$ (או $\frac{\partial L}{\partial b^{(l)}}$), ואז העדכון לפרמטרים יהיה

$$W_{i,j}^{(l) \text{ new}} = W_{i,j}^{(l) \text{ old}} - \eta \frac{\partial L}{\partial W_{i,j}^{(l) \text{ old}}}$$

איך זה מתבצע? (הערה – יש כל מיני אלגוריתמים שלא היה לי זמן לעשות את ההבחנה ביניהם – Gradient Descent, Stochastic Gradient Descent, Batched Gradient Descent ועוד. כאן מתואר הרעיון הכללי)

1. (forward) עוברים קדימה לאורך הרשת – בהינתן קלט \vec{x} , מחשבים את הערכים שיהיו בכל אחד מהנוירונים, את הפלט ואת ה-loss.

2. (backward=backpropagation) מחשבים לאחור, מהשכבה האחרונה עד לשכבה הראשונה, את הגראדיינטים של הפרמטרים, ואז מעדכנים את ערכיהם.

חישוב הגראדיינטים הוא למעשה ביצוע חוזר של נגזרות לפי כלל השרשרת:

$$L = L(f_n(f_{n-1}(\dots(f_1(x))))), y)$$

כאשר:

$$f_j(o^{(j-1)}) = g_j(W^{(j-1)} \cdot o^{(j-1)} + b^{(j-1)}) = g_j(z^{(j)}) = o^{(j)}$$

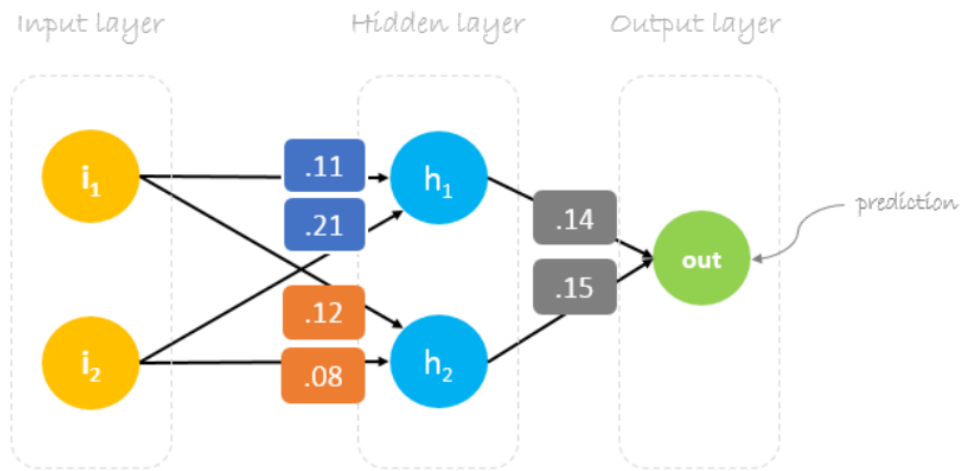
ו- g_j פונקציית אקטיבציה, ולכן (ייתכן שהחישוב האמיתי של זה ידרוש יותר סכומים. בכל מקרה זו לא הדרך הכי פורמלית להציג אותו, אבל זה עושה את העבודה):

$$\frac{\partial L}{\partial W_{i,j}^{(l)}} = \frac{\partial L}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W_{i,j}^{(l)}} = \frac{\partial L}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial o^{(l)}} \frac{\partial o^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W_{i,j}^{(l)}} = \frac{\partial L}{\partial z^{(l+1)}} W^{(l+1)} g'_l(o^{(l)}) \cdot o_j^{(l)}$$

הסיבה שהחישוב מתבצע לאחור הוא כי ניתן לשמור ערכים של נגזרות מכל שכבה עוקבת לשימוש בשכבה קודמת. כאן, זו הנגזרת $\frac{\partial L}{\partial z^{(l+1)}}$.

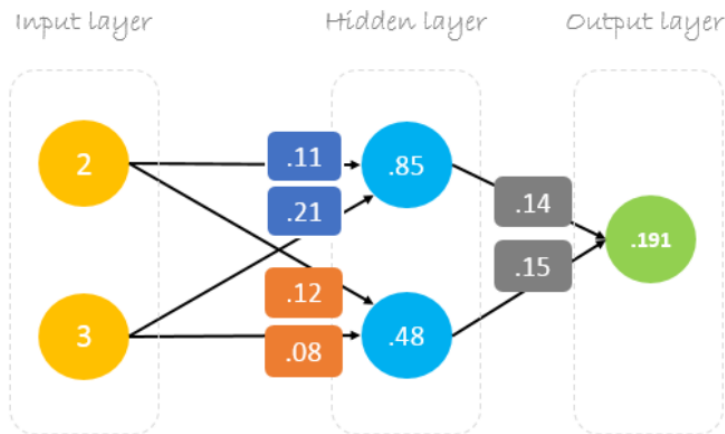
דוגמה (הקישור למקור):

נתונה הרשת הבאה (עם אתחול הפרמטרים הנתון, ובמקרה הזה ללא אקטיבציות וללא biases כלל):



נאמן את הרשת על הדגימה (2, 3), המתויגת 1, עם פונקציית ה-loss הבאה: $\frac{1}{2}(\hat{y} - y)^2$.

1. נלך קדימה לאורך הרשת, למציאת החיזוי שלה לערך הנקודה:



למשל, המעבר משכבת הקלט לשכבה החבויה:

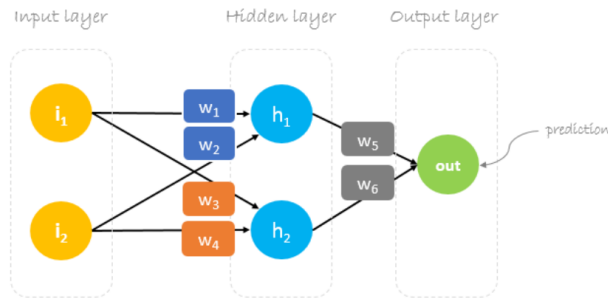
$$\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.11 & 0.21 \\ 0.12 & 0.08 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 0.85 \\ 0.48 \end{pmatrix}$$

השגיאה המתקבלת היא:

$$\text{Error} = \frac{1}{2}(0.191 - 1.0)^2 = 0.327$$

2. כעת, נבצע את ה-backpropagation כדי לשפר את הביצועים שלנו על הנקודה, ובמילים

אחרות – לאמן את המודל עליה:



מתחילים מהסוף – נחפש את הנגזרת החלקית של השגיאה לפי המשקלים האפורים.

למשל, עבור w_6 :

$$\frac{\partial \text{Error}}{\partial w_6} = \frac{\partial \text{Error}}{\partial \text{prediction}} * \frac{\partial \text{prediction}}{\partial w_6} \quad \leftarrow \text{chain rule}$$

$$\frac{\partial \text{Error}}{\partial w_6} = \frac{1}{2}(\text{prediction} - \text{actual})^2 * \frac{\partial ((i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6)}{\partial w_6}$$

$$\frac{\partial \text{Error}}{\partial w_6} = 2 * \frac{1}{2}(\text{prediction} - \text{actual}) * \frac{\partial (\text{prediction} - \text{actual})}{\partial \text{prediction}} * (i_1 w_3 + i_2 w_4)$$

$$\frac{\partial \text{Error}}{\partial w_6} = (\text{prediction} - \text{actual}) * (h_2)$$

$$\frac{\partial \text{Error}}{\partial w_6} = \Delta h_2$$

$\text{Error} = \frac{1}{2}(\text{prediction} - \text{actual})^2$
 $\text{prediction} = (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$
 $h_2 = i_1 w_3 + i_2 w_4$
 $\Delta = \text{prediction} - \text{actual}$ ← delta

ועבור השכבה הקודמת לה, למשל w_1 :

$$\frac{\partial \text{Error}}{\partial w_1} = \frac{\partial \text{Error}}{\partial \text{prediction}} * \frac{\partial \text{prediction}}{\partial h_1} * \frac{\partial h_1}{\partial w_1} \quad \leftarrow \text{chain rule}$$

$$\frac{\partial \text{Error}}{\partial w_1} = \frac{\partial \frac{1}{2}(\text{prediction} - \text{actual})^2}{\partial \text{prediction}} * \frac{\partial (h_1) w_5 + (h_2) w_6}{\partial h_1} * \frac{\partial (i_1 w_1 + i_2 w_2)}{\partial w_1}$$

$$\frac{\partial \text{Error}}{\partial w_1} = 2 * \frac{1}{2}(\text{prediction} - \text{actual}) * \frac{\partial (\text{prediction} - \text{actual})}{\partial \text{prediction}} * (w_5) * (i_1)$$

$$\frac{\partial \text{Error}}{\partial w_1} = (\text{prediction} - \text{actual}) * (w_5 i_1)$$

$$\frac{\partial \text{Error}}{\partial w_1} = \Delta w_5 i_1$$

$\text{Error} = \frac{1}{2}(\text{prediction} - \text{actual})^2$
 $\text{prediction} = (h_1) w_5 + (h_2) w_6$
 $h_1 = i_1 w_1 + i_2 w_2$
 $\Delta = \text{prediction} - \text{actual}$ ← delta

כאן הרשת קטנה, אז לא רואים יותר מדי שימוש ברקורסיה ובנגזרות שכבר חושבו, אבל

ברשתות עמוקות יותר השימוש הופך כמעט להכרחי.

לבסוף, אחרי שחישבנו את הנגזרות, נותר לעדכן את המשקולות (כאן נבחר קצב למידה של 0.05):

$$\Delta = 0.191 - 1 = -0.809 \quad \leftarrow \text{Delta} = \text{prediction} - \text{actual}$$

$$a = 0.05 \quad \leftarrow \text{Learning rate, we smartly guess this number}$$

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 0.85 \\ 0.48 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - \begin{bmatrix} -0.034 \\ -0.019 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - 0.05(-0.809) \begin{bmatrix} 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 0.14 & 0.15 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .08 \end{bmatrix} - \begin{bmatrix} -0.011 & -0.012 \\ -0.017 & -0.018 \end{bmatrix} = \begin{bmatrix} .12 & .13 \\ .23 & .10 \end{bmatrix}$$

אכן, כדי להיווכח שהמשקלים התעדכנו לכיוון הנכון, נסתכל ברשת עם הפרמטרים המעודכנים ונכניס אליה שוב את הנקודה. נקבל שהשתפרנו:

