חקר למידת

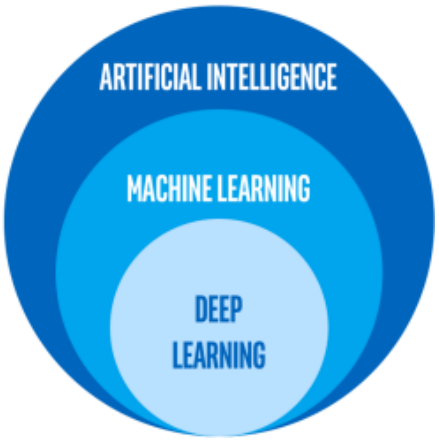
מכונה

רקע תיאורטי

בינה מלאכותית

בינה מלאכותית (או Artificial intelligence - AI), היא תחום נרחב במדעי המחשב (מתמטיקה, סטטיסטיקה והסתברות) שמטרתו העיקרית היא לגרום למכונות לבצע עבודה שרק בן אדם, לכאורה, יכול לעשות. לדוגמא: הבחנה בין כלב לחתול בתמונה. אם ננתח את הבעיה נראה שעל מנת לפתור אותה נצטרך 3 דברים: לדעת איך נראה כלב, לדעת איך נראה חתול ולהבדיל ביניהם. בעזרת סט הכלים שהיה עד עידן הבינה המלאכותית, לא היה ניתן "ללמד" את המחשב איך נראה כלב ולכן הבעיה הייתה בלתי פתירה.

הלב של הבינה המלאכותית הוא מבחן טיורינג, שמטרתו היא למדוד את רמת האינטיליגנציה של המכונה. במבחן, חוקר מתכתב עם שתי ישויות נסתרות מעיניו – אדם אחר ומכונה. אם נוצר מצב בו החוקר אינו יכול לקבוע מי מהם היה מחשב ומי מהם היה בן האדם, אזי שהמכונה הצליחה להתחקות אחר הבן אדם ויש לה אינטילגנציה שמחקה את זו של האדם.



כיום, קיימות דרכים שונות "ללמד" את המחשב דברים (כמו: איך נראה כלב) ובכך להשתמש בו כדי לבצע פעולות שעד לא מזמן נראו נחלתו של האדם. הענף של הבינה המלאכותית, למעשה עוסק ביכולת לתכנת מחשבים כך שיפעלו באופן שמייצג את הבינה האנושית.
שכפי שציינתי בינה מלאכותית היא תחום נרחב שכולל בתוכו את למידת המכונה כמו גם את הלמידה העמוקה – משפחות האלגוריתמים שמשמשים את המחשב בלמידה ועליהם אפרט בהמשך.

למידת מכונה

למידת מכונה (או Machine Learning - ML) היא יישום בפועל של בינה מלאכותית שמאפשר למערכת ללמוד ולהשתפר על סמך ניסיון בלי להיות מתוכנתת לכך במפורש. התהליך מתחיל באיסוף מידע – דגימות שמכילות דפוס מסויים על פיו יהיה ניתן לקבל החלטה בעתיד בהסתמך על דוגמאות מהעבר. ישנם שני סוגי בעיות עיקריים אותם מנסים לפתור בעזרת למידה מכונה: סיווג ורגרסיה. בבעיית הסיווג מחלקים את המידע למחלקות שונות ובחיזוי מנסים לגלות לאיזו מחלקה שייכת הדגימה. לעומת זאת בבעיית רגרסיה מנסים לייצר פונקציה שבהסתמך על כלל הפרמטרים תחזיר מספר.

שלבי סבב האימון

1. קבלת החלטה – מטרתו הסופית של המודל היא להחזיר סיווג / מספר ולכן הדבר הראשון שיעשה המודל הוא לנסות לקבל החלטה על סמך מה שאומן עד כה.
2. חישוב שגיאה – הערכה של החיזוי שהמודל ביצע בשלב הקודם במטרה לקבוע את הדיוק שלו.
3. ביצוע אופטימיזציה – עדכון המודל עפ"י תוצאת פונקציית השגיאה מה שמהווה בעצם את הלמידה.

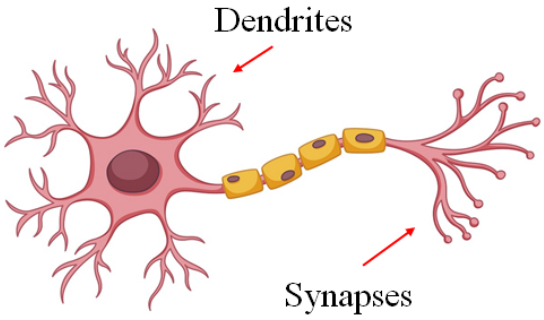
קבוצות אלגוריתמים של למידת מכונה

1. Supervised – האלגוריתם מוזן במידע מתוייג, כלומר, עבור כל דגימה יש תווית שמתארת אותה. על ידי למידה של הקשר בין הדגימות לתיוגיהן האלגוריתם יוכל לחזות מה יהיה התיוג של דגימה חדשה כלשהי.
2. Unsupervised – האלגוריתם מוזן במידע שאינו מתוייג ולכן מחפש בו תבניות ומוציא מהמידע שקיבל מידע נסתר. בסוג אלגוריתמים זה אין צורך בהתערבות של יד אדם והאלגוריתם לומד את המידע בעצמו. (קבוצה זו כוללת בין השאר את רשתות הנוירונים).
3. Reinforcement – קבוצת אלגוריתמים זו דומה לקבוצת ה-Supervised אך בניגוד אליה איננה מוזנת במידע מתויג, אלא מאומנת בצורה של ניסוי טעייה.

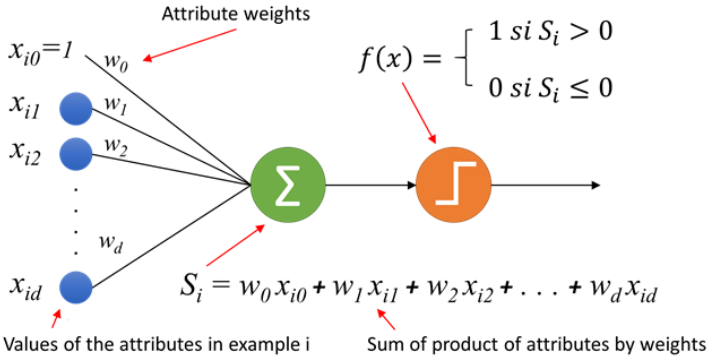
רשתות נוירונים ולמידה עמוקה

רשתות נוירונים הן קבוצה של אלגוריתמים שמנסים לזהות מערכות יחסים חבויות במערכים של מידע (Data Sets) בצורה המדמה את אופן פעולתו של המוח האנושי. בניגוד למוח שלנו, שמורכב מנוירונים אמיתיים, רשתות הנוירונים בנויות כמערכת גדולה של נוירונים מלאכותיים שמחקים את הנוירונים שבמוחנו. הייחודיות של רשתות הנוירונים היא ההסתגלות לקלט המשתנה בצורה דינמית, מה שמאפשר להן לייצר חיזויים מוצלחים ללא צורך בשינוי הרשת.

רשת הנוירונים בנויה משכבות של Node-ים, כשאר כל Node בה דומה לאלגוריתם הפשוט "Perceptron", שמדמה נוירון יחיד. ניתן לראות שגם במבנה שלו הפרספטרון דומה לנוירון אך הוא גם עובד בצורה זהה אליו. הרכיבים העיקריים של הנוירון האנושי הם: דנטריטים (הקולטנים של התא), גרעין התא, אקסון (שמעביר זרמים חשמליים לאורך התא) וסינפסות (שמעבירות זרים חשמליים לנוירונים האחרים שמחוברים לתא).



NEURON



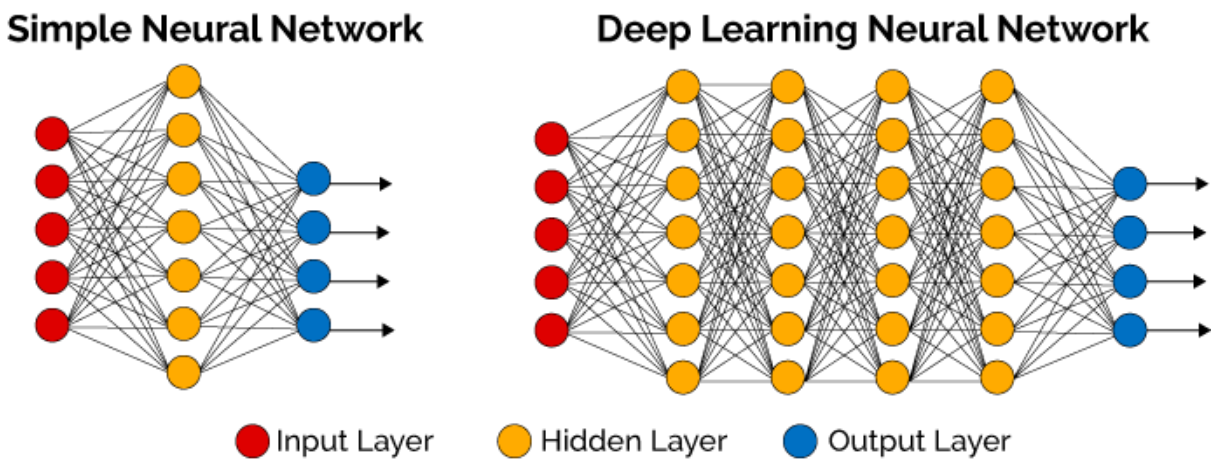
PERCEPTRON

בפועל, מה שקורה בנוירון אנושי הוא תהליך של קבלת מידע (זרם חשמלי) דרך הדנטריטים, עיבוד שלו והעברה הלאה דרך הסינפטות. בדומה אליו, עובד גם הנוירון המלאכותי - הפרספטרון.

הפרספטרון מקבל מידע, מבצע עליו פונקציה מתמטית (שמתבסס על הכפלת מטריצות – dot product) יחד עם המשקולות [החלק בתא ש-"לומד"], מעביר את הפלט דרך "פונקצית אקטיבציה" שמנרמלת את הערך (ראה הסבר בהמשך) ולבסוף מוציאה את הפלט מהתא.

כאמור, רשת הנוירונים בנויה משכבות של נוירונים מלאכותיים מעין אלו. בכל רשת נוירונים מחוייבות להיות שתי נקודות קצה – שכבת הקלט, ממנה הרשת מוזנת מידע ושכבת הפלט, דרכה הרשת מוציאה את תוצאת החישוב שלה. בנוסף, הרשת יכולה להכיל שכבות נוספות בין שכבת הקלט לשכבת הפלט, שנקראות "Hidden Layers". בין ה-Node-ים של שתי שכבות סמוכות קיימים חיבורים שמעבירים את הפלט של תא אחד לשני. לכל חיבור כזה יש "משקולת" – ערך, שכשמו,

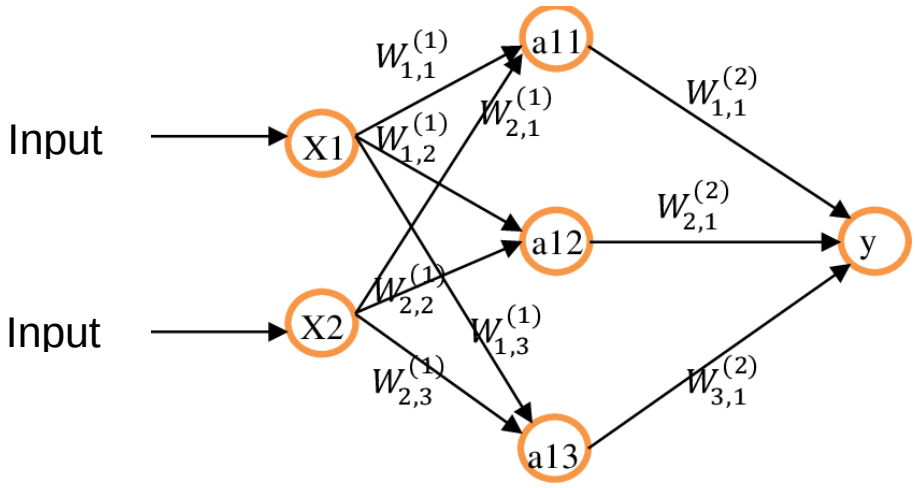
מחליט איזה משקל לתת למידע שעובר דרך החיבור. רשת מאומנת היטב היא אחת כזאת שכל המשקולות שלה מחזיקות בערכים האופטימליים ביותר להצלחת החיזוי.



רשת נוירונים שמכילה יותר מ-"Hidden Layer" אחת נקראת רשת למידה עמוקה. הנוירונים בשכבות אלו מבצעות שלל טרנספורמציות למידע שהרשת מקבלת כקלט ובעצם מייצרת לעצמה מאפיינים (features) חדשים שלא הופיעו במידע הגולמי. הרשת עושה תהליך של "Feature Extraction", הרשת לומדת לזהות אילו מאותם מאפיינים מהווים מרכיב משמעותי בלקיחת ההחלטה ובכך מעצמים את השפעתו של מאפיין זה, ואילו מנטרלים את ההשפעה של רכיבים אחרים. למעשה, הרשת מנצלת "תבניות נסתרות" במידע הבסיסי שהיא מקבלת על מנת לייצר את אותם מאפיינים. על ידי שילוב בין אותם מאפיינים גולמיים למאפיינים מתקדמים הרשת מצליחה להגיע להשיגים טובים יותר בלי התערבות יד אדם.

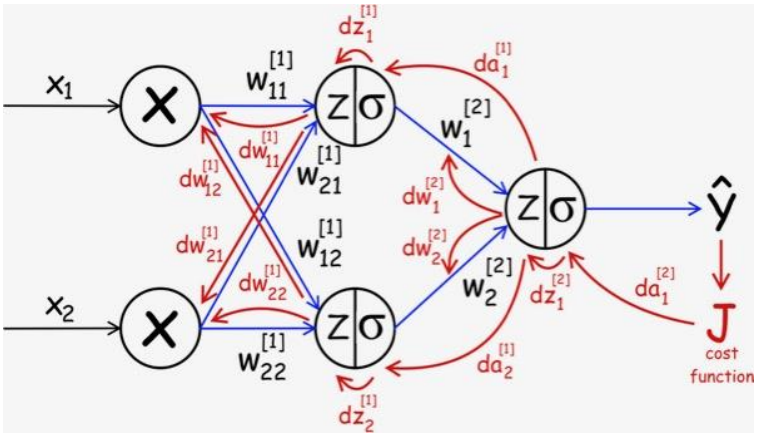
חלחול קדימה

חלחול קדימה (Forward Propagation) הוא התהליך של הזנת הרשת במידע וקבלת החיזוי. המידע שמוכנס לרשת עובר בין כל השכבות וכל Node מבצע עליו את הפעולה המתמטית שלו עד שלבסוף הרשת מתכנסת לשכבת הפלט שמחזירה את החיזוי. זה התהליך שעושים למידע אחרי שהרשת מאומנת כדי לקבל חיזויים, אך זהו גם חלק בלתי נפרד מהליך האימון של הרשת.



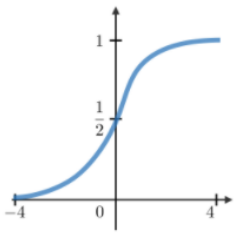
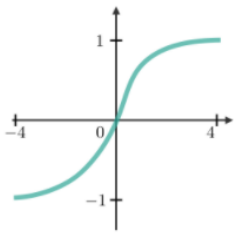
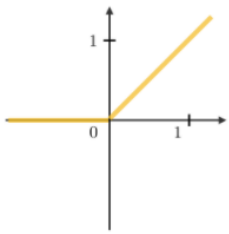
חלחול אחורה

חלחול אחורה (Backward Propagation) הוא התהליך שמאמן בפועל את הרשת ומעדכן אותה (את ערכי המשקולות שמרכיבות את הרשת). בשורה התחתונה, רשת נוירונים מבצעת הכפלה של מטריצות – מכפילה את הקלט במשקולות שלה. באימון, לאחר שנעשה חלחול קדימה והתקבל פלט שמייצג את מה שתצפה הרשת, נחשב את השגיאה בינו לבין הערך האמיתי, אותו ציפינו לקבל. לאחר קבלת השגיאה נחשב את הנגזרת שלה בהתייחסות לכל משקולת שיש ברשת. תהליך החלחול אחורה הוא בעצם "שרשרת של נגזרות" שמתחילה בשכבה האחרונה וחוזרת לשכבות הראשונות. לכל קשר מחשבים את הנגזרת שנקראת "gradient".נגזרת זו תשמש בהמשך לחישוב הנגזרת של של השכבות שלפניה. תהליך חישוב הנגזרות חוזר על עצמו עד שמתקבלת נגזרת לכל משקולת. לבסוף מחסרים מכל משקולת את הערך הנגזרת שלה על מנת להקטין את השגיאה שלה.



פונקציות הפעלה (Activation Functions)

פונקצית אקטיבציה היא פונקציה שמטרתה להוציא את התוצר מהתא. הרשת משתמשת בפונקציות אקטיבציה כדי לקבוע את רמת החשיבות של מידע על ידי העברתו לסקאלה אחרת מוגדרת, בדרך כלל בין 0 ל-1 או בין (-1) ל-1. פוקציות האקטיבציה הנפוצות ביותר הן Sigmoid, Tanh ו-ReLU. לכל פונקציה יש אופן חישוב שונה, התנהגות שונה וטווח ערכים שונה, ולכן לכל אחת יש את השימוש הייחודי לה.

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

מושגים נוספים

פונקציות מחיר – משפחת פונקציות שנועדו למדוד את השגיאה של המודל. לכל סוג בעיה (סיווג / רגרסיה) יש פונקציות מחיר שונות שמתבססות על פרמטרים שונים.

שיעור למידה (Learning Rate) – המשתנה שקובע כמה מהר / לאט יתעדכנו המשקולות. בחלחול לאחור המשקולות מתעדכנות על ידי חיסור הנגזרת מהערך האמיתי שלהם, שיעור הלמידה קובע את מידת ההשפעה של חיסור ערך הנזרת. שיעור למידה גבוה יעזור לרשת ללמוד מהר יותר אך יכול לגרום למצב של פיספוס הערכים האופטימליים למשקולות.

Batch Size – מספר הדגימות שנכנסות למודל בסבב אחד של חלחול קדימה / אחורה.

Training Epochs – כל epoch מייצג מעבר שלם של חלחול קדימה ואחורה על כל סט הנתונים. סך כל ה-epochs מייצג את מספר הפעמים שהמודל נחשף למידע האימון ולמד אותו.

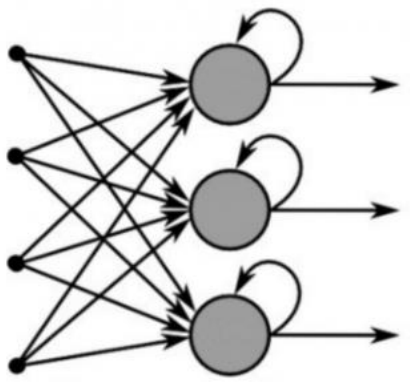
רשתות נוירונים מסוג RNN

הקדמה

Recurrent Neural Networks היא משפחה של רשתות נוירונים שיודעות להתמודד עם רצפים (sequence) של מידע שמתקבלים כקלט. מה שמאפשר לרשתות אלו להתמודד עם רצפי המידע הוא הזיכרון הפנימי של הרשת שמכיל מידע חשוב על הקלטים הקודמים שהתקבלו. תכונה זאת היא שהופכת את רשתות ה-RNN לכה מדוייקות ומתאימות לתחומים: דיבור (לדוגמא: עוזרות קוליות), טקסט, אודיו, וידאו, מזג האוויר, סדרות זמן וכמובן מידע פיננסי. בהשוואה לאלגוריתמים אחרים, ל-RNN ישנה ההבנה של חשיבות הרצף וההשלכות שלו על החיזוי. למעשה, אפשר להגיד שרשתות RNN מתאימות לפתירת בעיות בהן חשיבות הרצף עולה על חשיבותו של frame בודד בו.
[רצף מידע – מידע שמסודר בסדר כרונולוגי, על פי תאריכים או סדר אירועים. וזה הייחוד של RNN לעומת רשת נוירונים רגילה שהקלט שלה הוא מידע שאין חשיבות לסדר שלו.]

אופן העבודה של רשת RNN

בניגוד לרשת הנוירונים הבסיסית ("Feed-Forward"), שכל Node בה רק מעביר הלאה ערך, ב-RNN ה-Node גם מחזיר לעצמו ערך שישמש את עצמו בהמשך. שמירת הערכים הזו היא שמאפשרת לרשת ללמוד את הרצף, ולא רק להתייחס לנקודת זמן אחת בו.
בפועל, יוצא שה-Node מעבד שני קלטים – הקלט החדש (batch) וה מצב הקודם שהוא היה בו, שנקרא hidden state.

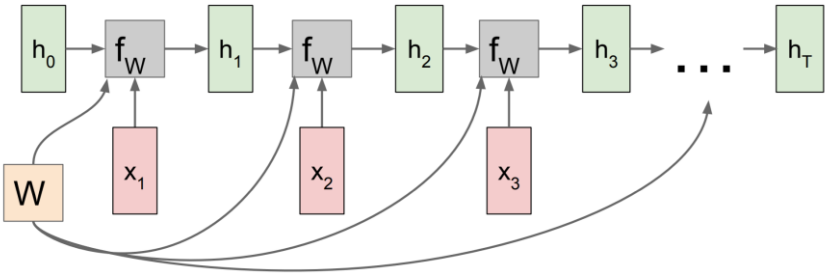


Recurrent Neural Network

בתהליך האימון התא מפעיל פונקציה כלשהיא על ה-hidden state הקודם של התא ועל הקלט הנוכחי יחד. התוצאה של פונקציה זו תהווה את המצב החדש של התא שיועבר אליו עם כניסת הקלט החדש.

$$h_t = f_W(h_{t-1}, x_t)$$

לאורך הזמן הרשת מכוונת את משקולותיה ולומדת אילו ערכים עליה לשמור ואילו ערכים כדאי "לזרוק", ההתייחסות ל-hidden state היא זו שלמעשה מאפשרת את ההסתמכות על הרצף (המצב הקודם של התא מושפע מהקלט שקיבל בעברו). כך שלבסוף, הרשת יודעת לחזות על סמך העבר והשינוי שהתרחש לאורכו את המשך הרצף.



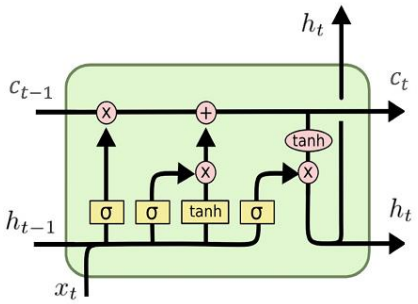
מודלים בפרוייקט

ארכיטקטורת LSTM

הבעיה ש-LSTM פותר

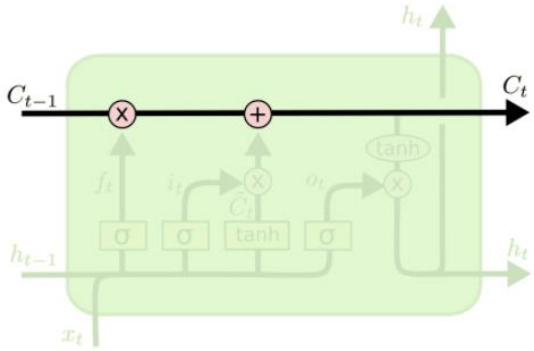
באימון רשת מסוג RNN המאפשרת למידה של רצפים (sequences) יכולה להווצר בעיה שלאורך זמן תאבד ההשפעה של החלק הראשון של המידע. למשל: כשמעבדים פסקה של טקסט, רשת ה-RNN עלולה להשמיט משפטים מתחילת הפסקה. בעיה זאת, שנוצרת בזמן החלחול לאחור, נקראת "vanishing gradients" ומייצגת מצב בו ה-gradients הופכים להיות קטנים עד כדי איבוד ההשפעה מה שגורם להפסקת הלמידה.

ארכיטקטורת Long-Short Term Memory נועדה לפתור בעיה זאת. הארכיטקטורה מותאמת ללמידה לאורך פרק זמן ארוך. כל תא LSTM יכול לקרוא, לכתוב, לשנות ולמחוק את הזיכרון שלו. את השינויים בזיכרון עושה התא באמצעות ה-"שערים" שבו, שבעצם לומד עם הזמן איזה מידע חשוב ואיזה לא (ולפי חשיבות המידע הוא "פותח" / "סוגר" את השער). לתא LSTM יש שלושה שערים: Forget Gate, Input Gate ו-Output Gate. באופן כללי ה-Input Gate אחראי על הכנסת/חסימת מידע חדש, ה-Forget Gate אחראי על מחיקת מידע שכבר לא חשוב יותר וה-Output Gate אחראי על רמת ההשפעה של זיכרון התא על הפלט.



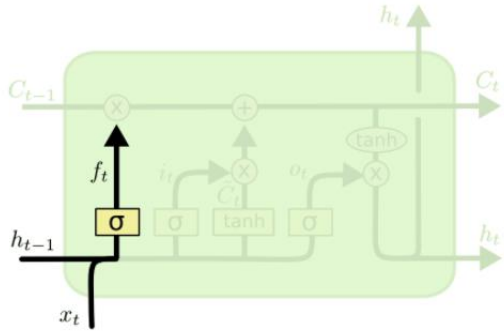
LSTM
(Long-Short Term Memory)

Cell State



המפתח לפתרון ש-LSTM מציע הוא ה-"Cell State".
ניתן לראות אותו בחלק העליון של מבנה התא.
ה-Cell State שומר בקביעות על המצב הקודם של התא ויכול לבצע בו שינויים (מינוריים יחסית) על ידי הכפלה עם התוצר של ה-Output Gate וסכום עם התוצר של Forget Gate. אבל בעיקרון חלק זה אחראי על הזיכרון לטווח ארוך וכמעט ולא מושפע משינויים של frame ספציפי.

Forget Gate

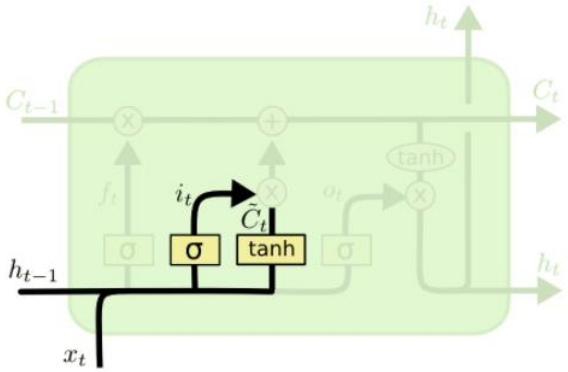


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

השלב הראשון שקורב ב-LSTM הוא להחליט איזה מידע צריך להיזרק מה-Cell State. ההחלטה נעשית על ידי שכבת סיגמואיד שמקבלת את ה-hidden state הקודם (h_{t-1}) ואת הקלט הנוכחי (X_t) ומחזירה מספר בין 0 ל-1 עבור כל מספר ב-Cell State. 1 מייצג שמירה מלאה של המידע ואילו 0 מייצג "זריקה" של המידע (עבור כל מספר ב-Cell State מתבצעת הכפלה במספר התואם שלו שמוציא הסיגמואיד, הכפלה ב-1 שומרת על המפסר כמו שהוא, הכפלה ב-0 מאפסת אותו).

[שימוש לדוגמא: כאשר מנתחים משפט ומגיעים לנושא חדש, יש לשכוח את המגדר של הנושא הקודם].

Input Gate

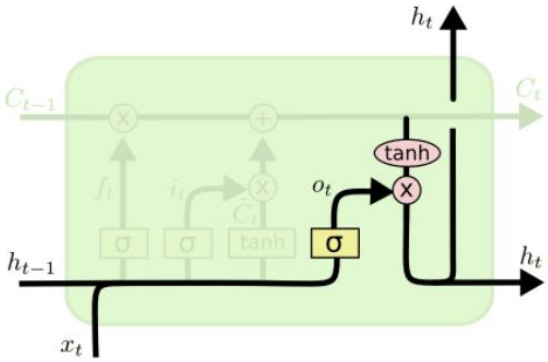


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

השלב השני הוא בחירת המידע שנרצה לאחסן ב- Cell State. שלב זה מורכב משני חלקים – הראשון שכבת סיגמואיד, כמו ב-Forget Gate שנועד לבחור ערכים אותם נרצה לעדכן. השני, שכבת tanh שיוצרת וקטור של ערכים חדשים שיוכלו להתווסף ל-Cell State. הכפלה בין התוצאות של שני החלקים תיצור את המידע שנוסיף לאחסון ב-Cell State.

Output Gate



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

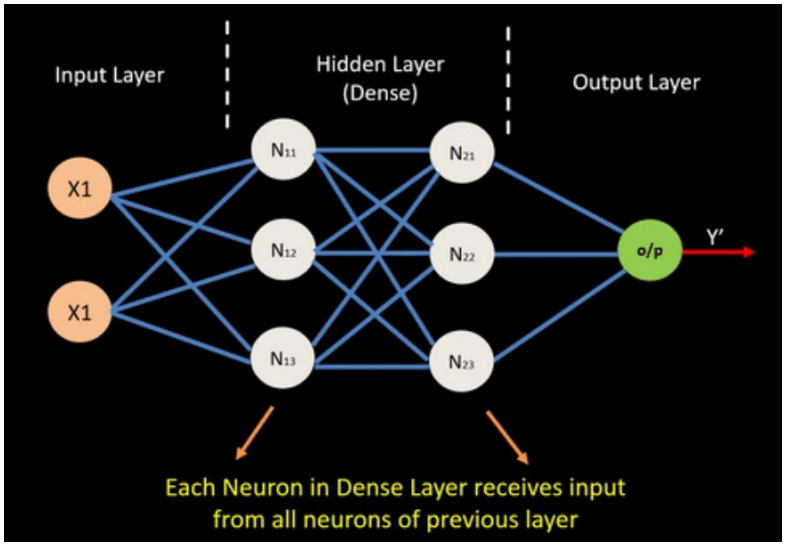
$$h_t = o_t * \tanh\left(C_t\right)$$

השלב האחרון הוא בחירת המידע שנרצה לייצא. התוצר יהיה מבוסס על ה-Cell State, אך יעבור מניפולציה עם המידע החדש. גם שלב זה מחולק שלני חלקים – הראשון, שכבת סיגמואיד (כמו בשערים הקודמים). השלב השני הוא העברת ה-Cell State דרך שכבת tanh (שתעביר את הערכים להיות בסקאלה של (-1) – 1). הכפלה בין התוצאות של שני החלקים תיצור את המידע שייצא התא – הפלט שלו וגם ה-hidden state החדש.

[שימוש לדוגמא: כאשר מנתחים משפט ומגיעים לנושא חדש, יש להעביר הלאה מידע אודות הנושא, כמו יחיד / רבים].

שכבת Dense

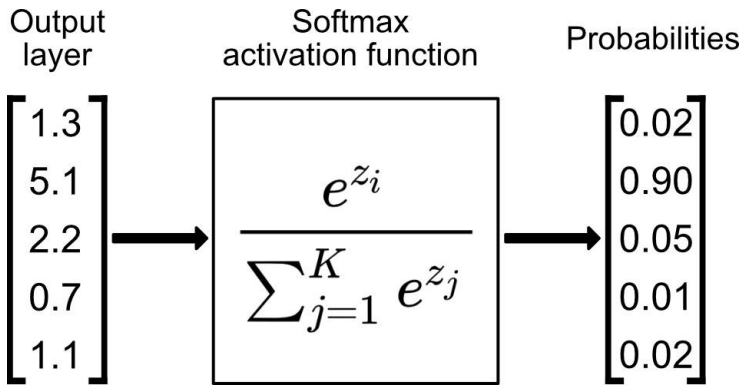
שכבת Dense משמשת כגורם המקשר במודל, היא נקראת גם "Fully connected" מכיוון שהיא מחברת את החלקים במודל בצורה צפופה כמו רשת. כל תא בשכבה זו מוזן מכל הפלטים של השכבה שלפניו. המטרה של שכבה זו היא לייצר מאפייני למידה "Feature Extraction" מכל הקומבינציות האפשריות של הקלטים של השכבה (בניגוד ל-CNN שמתבסס על המאפיינים בשדה קטן יחסית). השכבה יוצרת בעצם אופרציה לינארית על השכבה שלפניה בה יוצרים (מספר_התאים_בשכבה_הקודמת) * (מספר_התאים_בשכבה_הנוכחית) קשרים. בגלל שנוצרים כל כך הרבה קשרים, בדרך כלל אחרי שכבת ה-Dense תבוא שכבה לא לינארית שתבחר את המאפיינים המוצלחים יותר שנוצרו.



פונקציית SoftMax

פוקציה זו מקבלת כקלט וקטור בגודל K של ערכים מספריים (חיוביים, שליליים או 0) ומחזירה וקטור בגודל K שסכום כל הערכים בו שווה ל-1. הפונקצייה ממירה את כל הערכים בוקטור המקורי לערכים בתחום 1 – 0 שמבטאים את ההתסברויות. פונקציה זו משמשת בדרך כלל בסיווג מסוג multi-class בו מחשבים את ההסתברות שהקלט מתאים לכל אחת מהמחלקות. התיוג שקיבל את הערך הכי גבוה מבטא את הסיווג של המודל.

[הפונקציה מאוד דומה בהתנהגודה לסיגמואיד, אך בעוד שסיגמואיג משמש לסיווג בין שתי מחלקות, softmax משמש לסיווג בין מספר רב יותר של מחלקות].



הנתונים

One Hot Vector

כאשר קיימת בעיית סיווג לקטגוריות, והתיוג של המידע הוא איננו מספרי, הרבה מודלים יתקשו להתמודד עם התיוג. המודלים דורשים שכל ערכי התיוג והחיזוי יהיו מספריים ולכן עולה צורך להמיר תיוג מילולי לתיוג מספרי שיהיה ניתן להמיר מאוחר יותר בחזרה לתיוג מילולי.

על מנת להתמודד עם בעיית התיוג המילולי ניתן לשייך לכל מחלקה מספר ייחודי שיזהה אותה.

אך כאשר נצטרך לפתור בעיית סיווג מסוג multi-class המודל יצטרך לקבל כקלט וקטור של תיוגים.

One Hot Vector הוא וקטור שמכיל אלמנטים כמספר הקלאסים השונים, כאשר כל הוקטור מלא באפסים מלבד איבר אחד שהוא 1. עבור כל תיוג ה-1 הוא איבר אחר בוקטור וכך ניתן לתאר סוגים שונים של מחלקות באמצעות וקטור מספרי בצורה פשוטה.

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

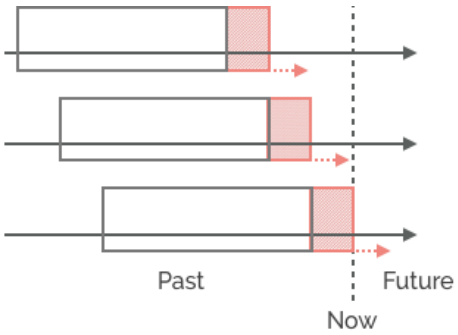
Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Sliding Window

כאשר מתעסקים עם נתונים שמבוססים על Time Series לפעמים נרצה לנצל את את הרצף שיוצרים הנתונים כדי לנסות לחזות את המשך הסדרה. לכן, יש צורך בארגון מחדש של המידע כך שכל קלט (X) יהיה מורכב מנתונים של מספר ימים אחורה והתיוג (y) יהיה הנתון של היום הבא.

מספר הימים (או פרקי הזמן) שנכנסים ב-X נקרא "גודל החלון". Sliding Window הוא דרך של ארגון מחדש של המידע, כך שכל n פרקי זמן ירכיבו חלון והערך של פרק הזמן n+1 יהיה התיוג שלו.

כל Frame של מידע מכיל מידע חופף ל-Frame שלפניו וכך נוצר מבנה של חלון זז. ישנה שיטה להשתמש ב-Sliding Window כדי לחזות כמה שלבים קדימה, ובדרך זאת בכל חיזוי של שלב קדימה מסתמכים על הפלט של המודל בשלב האחרון.



Scaler

Scaling הוא תהליך המרה שעושים למידע מספרי כך שיכנס בתחום מסויים של ערכים.
כאשר משתמשים ב- Scaler, ה-Scaler מקבל את המידע ואת טווח הערכים, מוצא את הערכים המקסימליים במידע וקובע אותם לקצוות של טווח הערכים. בעזרת פונקציה מתמטית ה-Scaler מתאים את שאר המידע שקיבל לערכים בתחום הנ"ל. הסיבה שעורכים Scaling למידע היא כדי להקטין את המספרים שהרשת מתעסקת איתם, שכן מספרים גבוהים יותר גורמים לתהליך החישוב להיות מסובך וארוך יותר. יש להבדיל בין Scaling ובין Normalization, נורמליזציה היא שלב מתקדם יותר של Scaling בו על המידע מופעלת פונקציה שתגרום לפיזור שלו בצורה סטטיסטית (כמו "bell curve").

Yahoo Finance

להורדת נתונים על מניות קיימים מספר מקורות. מרבית ה-APIs שמאפשרים להוריד מידע פיננסי מגבילים את כמות הקריאות ל-API לקריאות בודדות בפרק זמן נתון (כמו Alpha Vantage). למרות העבודה ש-APIs אלו חוסכים בניתוח המידע, הפרוייקט לא יכול להתבסס על הגבלת קריאות.
בפייתון קיימת חבילה של Yahoo Finance שמאפשר הורדת נתונים היסטוריים על מניות בצורה פשוטה ונוחה ללא הגבלת קריאות.

דיון

כאשר עוסקים ב-Data Science, מנסים להוכיח שטענה כלשהי מתקיימת בהתבסס על נתונים ודגימות רבות או במילים אחרות "Given X predict Y".
בתחילה, ניסתי להוכיח את הטענה "בהינתן מידע היסטורי של מניה מסוימת 'n' ימים אחורה, צפה את שער הסגירה של המניה ביום הבא".

מטענה זאת עולות הדרישות הבאות:

1. נדרש ניתוח של כמות גדולה של מידע.
2. יש חשיבות לרצף המידע.

מהדרישה הראשונה ניתן להסיק שהאלגוריתם המתאים לפתרון הבעיה הוא רשת נוירונים (ואולי אפילו רשת של למידה עמוקה). מהדרישה השניה ניתן להסיק שרשת הנוירונים צריכה להיות מסוג RNN, מכיוון שיש חשיבות רבה לרצף הנתונים על המניה ב-n הימים האחרונים.
מכיוון שבטענה דובר על לצפות את שער הסגירה, שהוא נתון מספרי הבעיה היא בעית רגרסיה.
עם זאת, בעיה מסוג זה היא קשה יותר לפתירה מאשר בעיית סיווג ולכן שיניתי את טענת המחקר ל-"בהינתן מידע היסטורי של מניה מסוימת "n ימים אחורה, צפה את המגמה בשער הסגירה ליום הבא (עלייה / ירידה / סטטיות)".

לאחר סקירת כמה מקורות, שלא איפשרו לי להוריד את המידע שאני צריך בצורה חלקה (כדוגמת AlphaVantage), בחרתי בספרייה של yahoo finance כמקור המידע שלי. מקור מידע זה מספק את הנתונים הבאים: Open, Close, High, Low, Volume.
בתחילה ניסתי לאמן את המודל רק על נתוני הסגירה, כפי שראיתי במקורות שונים, אך לא הצלחתי כמותם. לאחר זמן רב של מחקר גיליתי שהבעיה הייתה נעוצה באופן השימוש ב-Scaler. בעוד שהדרך הנכונה היא לקבוע ל-Scaler גבולות על פי נתוני האימון בלבד, באותם מקורות קבעו את הגבולות על פי כלל המידע, מה שגרם לחיזוי להיות שיקרי ומוטעה.

נוצר מצב שלא הצלחתי לחזות בעזרת נתוני הסגירה בלבד ולכן עברתי להשתמש בכל הנתונים שהורדתי.גם ניסיון זה לא צלח ובמשך זמן רב ניסיתי לחפש האם הבעיה היא במודל עצמו או בנתונים. לאחר בחינת הנתונים על מודל בסיסי (SVC שהוא SVM לבעיית סיווג), ובחינת מידע בסיסי על המודל שלי, הגעתי למסקנה שהבעיה היא בנתונים – חסרים מאפיינים.

לצורך ייצור נתונים חדשים "Feature Extraction" השתמשתי ב-TALIB שהיא חבילה שנועדה לבצע ניתוח ואנליזות טכניים. חקרתי על מספר מדדים טכניים שיכולים לשמש אותי כמאפיינים חדשים והוספתי אותם לסט הנתונים.

על כלל הנתונים ביצעתי תהליך של preprocessing. ראשית, ניקתי מהמידע שורות חסרות ורעשים. בגלל שבחרתי לפתור בעית סיוג נצרכתי לתייג את המידע ועשיתי זאת בשיטת One Hot. לאחר מכן עשיתי למידע תהליך של Scaling, כאשר ה-Scaler קבע את הגבולות לפי נתוני האימון בלבד (כמו שצריך!). לבסוף, שיניתי את מבנה המידע לצורה של Sliding Window כך שתהיה משמעות לרצף המידע, שכן, כפי שכתבתי, ישנה חשיבות לרצף המידע. המידע גם חולק לסט אימון וסט בדיקה ביחס 80-20 בהתאמה [ומסט האימון 20% שימשו לואלידציה].

המודל עצמו הוא רשת מסוג RNN (לצורך ההתמודדות עם רפצי המידע), ונעשה בה שימוש בארכיטקטורת LSTM. ה-LSTM דואג להתייחס גם לנתונים מתחילת ה-batch וגם לנתונים מסופו ולכן יתאים במיוחד לבעייה בה רצף המידע הוא של ימים רבים. מלבד שכבות ה-LSTM, המודל מורכב גם משכבות Dense ו-softmax. השכבה הראשונה היא Dense כדי לנסות לייצר כמה שיותר מאפיינים. לאחר מכן מגיעות שכבות ה-LSTM שעושות את עיקר העבודה ומיד לאחריהן יש שכבת Dense של שלושה Nodes מכיוון שהמידע מסווג לשלוש מחלקות. לבסוף, שכבת ה-softmax קובעת את ההסתברות של הדגימה להשתייך לכל אחת מהמחלקות וזה גם המידע שנשלח למשתמש.

YAHOO! FINANCE DATA						
Date	Open	High	Low	Close	Volume	Adj Close
11/01/16	$59.97	$60.02	$59.25	$59.80	24,533,000	**$59.40**
11/02/16	$59.82	$59.93	$59.30	$59.43	22,147,000	**$59.03**
11/03/16	$59.53	$59.64	$59.11	$59.21	21,600,400	**$58.81**
11/04/16	$58.65	$59.28	$58.52	$58.71	28,697,000	**$58.32**
11/07/16	$59.78	$60.52	$59.78	$60.42	31,664,800	**$60.01**
11/08/16	$60.55	$60.78	$60.15	$60.47	22,935,400	**$60.06**

Preprocessing – clean, scale, sliding window and split to train and test