שיעור 6

22 במאי 2019

1 Recurrent Neural Networks

בשיעור זה נחקור פתרונות ללמידה של סדרות. בתרחישים אלו, המידע שלנו מורכב מסדרות, עם חשיבות לסדר בתוכן. ישנן דוגמאות רבות למצבים שכאלו:

- עבודה עם שפה טבעית: למשל, נרצה לחזות לרצף מילים את המילה הבאה במשפט. מודל שכזה נקרא מודל שפה.
- חיזוי אותות: למשל, נרצה לחזות את הערך של מניה על־סמך הערך שלה בעבר.
- מענה על שאלות: עבור שאלה המוזנת בשפה טבעית, נרצה להחזיר תשובה לשאלה.
- עבור סדרת מספרים, נחזה את המספר הבא בסדרה.

אילו קשיים יש בבעיות מסוג זה? אנו עשויים לפגוש סדרות ארוכות מאוד. אם נרצה לטפל בהן ברשת נוירונים fully connected נגלה כי דרושים לנו פרמטרים רבים לשם־כך. בנוסף, פתרון שכזה לא ינצל קשרים טבעיים שהיינו מצפים שיהיו בין איברים בסדרה. כמו בתמונות, גם כאן ישנן תכונות "לוקאליות" שניתן לצפות שהסדרה תקיים, ושהיינו רוצים להשתמש בהן לחיזוי.
קושי נוסף שאנו עשויים לפגוש הוא בעבודה עם סדרות באורכים משתנים: כזכור, ברשת fully connected אנו נדרשים להגדיר את גודל השכבות. איך היינו עשויים לרצות להתגבר על כך? היינו יכולים להגדיר מראש גודל קלט גדול מספיק ולהשתמש, במידת הצורך, ב־padding: ריפוד של הקלט בקבועים (למשל אפסים) כדי להתאים אותו לגודל הקלט המוגדר. עם זאת, פתרון שכזה רק יגדיל את מספר המשקולות ברשת ויקשה על אימונה. נחפש ארכיטקטורה של רשת נוירונים שתאפשר לנו להתמודד עם הבעיות הללו.

1.1 מקרה בוחן

נפתח כלים לעבודה על מידע סדרתי אל מול מקרה הבוחן של מודל שפה: נרצה לבנות מודל שבהינתן אוסף מילים, חוזה את המילה הבאה במשפט. למשל, עבור המשפט once upon a time, in a land far far, נרצה שהמודל יחזה את המילה away. נהוג לחשוב על בעיות שכאלו כעל בעיות classification, כאשר המודל חוזה אחת מתוך המילים במילון. לכן עשויות להיות לנו עשרות או מאות אלפי קטגוריות. המודל יתן לכל אחת מהקטגוריות הסתברות לכך שהיא המילה הבאה במשפט. המילה עם ההסתברות הגדולה ביותר תבחר.
כיצד מוודאים כי החיזוי של המודל אכן מייצג הסתברות? עלינו לוודא כי סכום ההסתברויות

שהמודל חוזה אכן נסכם ל־1. לכן נהוג לבצע נרמול על התחזיות באמצעות פונקציית softmax: אם התחזיות של המודל הן $z_1, ..., z_n$ אזי

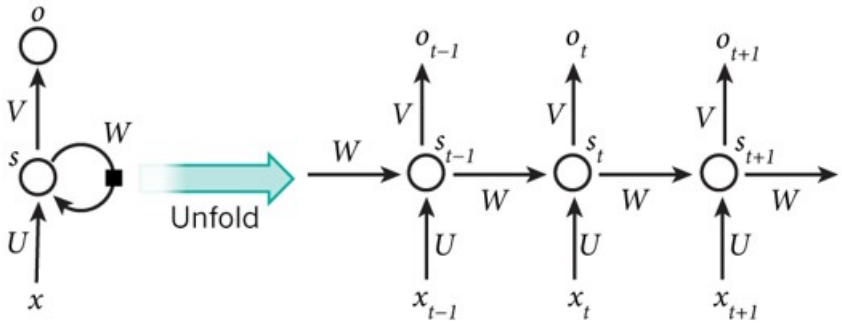
$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

שימו לב כי אכן סכום התוצאות המתקבלות הוא 1. softmax היא קירוב חלק של פונקציית ה־argmax, שמחזירה 1 לערך המקסימלי ו־0 לאחרים. כלומר, זו פונקציה גזירה, ונוכל להשתמש בה כפונקציית האקטיבציה האחרונה ברשת multi labels, כלומר עם כמה קטגוריות.
כיצד נוכל לעבוד עם המילים באלגוריתם למידה? איזה תהליך feature extraction נוכל להפעיל על משפט? למעשה זו שאלה קשה למדי, ונתעמק בה יותר בשיעור הבא. בינתיים נסתפק בפתרון פשוט: 1 hot encoding: אם המילון שלנו הוא בגודל N, נקודד את המילה ה־i לוקטור בגודל N, עם 1 במקום ה־i ו־0 במקומות האחרים. נקבל וקטורים ארוכים מאוד, דבר שלרוע המזל יבטיח כמו פרמטרים רבה ברשת.
כעת אנו מחפשים ארכיטקטורה שתקבל רשימה מסודרת של וקטורים באורך N, $u_1, ..., u_n$, ותחזה עבורן התפלגות על המילון.

1.2 ה־RNN הפשוט

נגדיר שכבת RNN (Recurrent Neural Network) פשוטה. ב־RNN נרצה לבצע פעולה כלשהי על כל קלט בסדרה ו"לזכור" את הקלטים הקודמים. נוכל לחשוב על מבנה שכזה כעל רשת עם "זכרון". השרטוט הבא מגדיר את אופן פעולת השכבה:

איור 1:



נחשוב על הקלט כעל סדרה התלויה בזמן. x_t הוא הקלט בזמן t. U, W, V הן מטריצות משקולות הקשורות לשכבה. פעולתה מוגדרת כך:

$$s_t := \sigma(U \cdot x_t + W \cdot s_{t-1})$$
$$o_t := V \cdot s_t$$

s_t הוא המצב החבוי, ה־hidden, של הרשת, בזמן t. הוא תלוי ב־x_t, הקלט בזמן t, וב־s_{t-1}, ה־hidden בזמן הקודם. בהינתן סדרת הקלטים נוכל לחשב את $s_1, ..., s_n$ ולקבל מכך רשימה של פלטים, $o_1, ..., o_n$.

U, W, V הן, כאמור, מטריצות המשקולות של השכבה. נשים לב ראשית כי אנו משתמשים בכמות קטנה ביותר של משקולות, ביחס לכמות שניתן היה לצפות שנדרש לה. המימד של הוקטורים s_t, o_t נקבע על־ידנו בהגדרת השכבה, ולכן

$$U \in \mathbb{R}^{dim(s)\times dim(x)}$$
$$W \in \mathbb{R}^{dim(s)\times dim(s)}$$
$$V \in \mathbb{R}^{dim(o)\times dim(s)}$$

נחשוב על U, W כעל מטריצות ש"מתווכות" את הקלט ואת ה־hidden הקודם כדי לקבל את ה־hidden החדש. היינו רוצים לחבר באופן כלשהו את הקלט החדש עם המידע שצברנו ממעבר על קלטים קודמים. הדרך הטבעית לחבר ביניהם היא באמצעות כפל במשקולות, וחיבור. חיבור פשוט אינו הדרך הטבעית: תוכלו לחשוב על כך כעל חיבור של גדלים מיחידות שונות. בראייה זו, U ו־W "מסדרות לנו את היחידות במשוואה".
σ היא פונקציית אקטיבציה, לרוב לא ליניארית - למשל, פונקציית הסיגמואיד שהכרנו. V היא הדרך שלנו לעבור מה־hidden לפלט. פעמים רבות נהוג להשמיט אותה.
נתעכב על השרטוט: בצורתו ה־unfolded הוא מתאר את מעבר הקלטים ברשת. נהוג להשתמש בשרטוטים "מקופלים", במיוחד כשהרשתות מסתבכות. בשרטוטים שכאלו, כל node מייצג וקטור שעלינו לחשב. החצים הנכנסים אליו מתארים באילו וקטורים אחרים עלינו להשתמש כדי לחשב אותו, כאשר המשקולות ש"מתווכות" את הגדלים הללו מצוינות על החצים.
את s_t נוכל לפרש כ"זכרון" של השכבה: הוא מתעדכן עם כל קלט חדש, אך מכיל "אינפורמציה" מהקלטים הקודמים. עם זאת, ישנה הגבלה על כמות ה"מידע" שניתן לאחסן בו.
הערה: בביטויים ל־s_t ו־o_t נהוג ואף מומלץ להוסיף רכיב של bias שהתעלמנו ממנו כאן, ממש כפי שעושים ברשתות fully connected רגילות.

1.2.1 אימון ו־backpropagation

נהוג להגיד ברשתות שכאלו כי ישנו שיתוף פרמטרים בין כמה מטריצות: זאת כיוון ש־U, W, V מופיעות כמה פעמים בשרטוט. אנו לא נחשוב באופן זה, כיוון שאנו מכירים את כלל השרשרת: אם אנו מעוניינים, למשל, לחשב את $\frac{\partial L}{\partial w_{ij}}$, כל שעלינו לעשות הוא להשתמש בכלל השרשרת כדי לקבל את הנגזרת הזו. באופן טבעי נקבל את הנגזרת "האמיתית", שמתחשבת בכל הפעמים בהם השתמשנו ב־W ברשת. נדגים במקרה הקל, בו W, U, V הן מטריצות 1×1:

$$\frac{do_{t+1}}{dU} = \frac{\partial o_{t+1}}{\partial s_{t+1}} \cdot \frac{ds_{t+1}}{dU} = \frac{\partial o_{t+1}}{\partial s_{t+1}} \cdot \left(\frac{\partial s_{t+1}}{\partial U} + \frac{\partial s_{t+1}}{\partial s_t} \cdot \frac{ds_t}{dU}\right) =$$
$$\frac{\partial o_{t+1}}{\partial s_{t+1}} \cdot \left(\frac{\partial s_{t+1}}{\partial U} + \frac{\partial s_{t+1}}{\partial s_t} \cdot \left(\frac{\partial s_t}{\partial U} + \frac{\partial s_t}{\partial s_{t-1}} \frac{ds_{t-1}}{dU}\right)\right) = ...$$

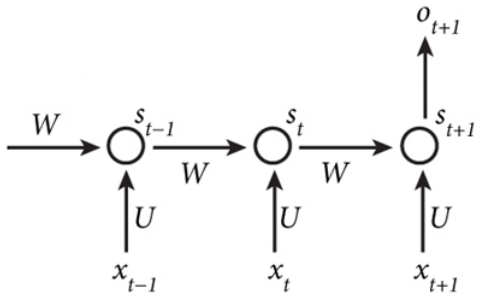
כעת, ה־Loss של הרשת הוא פונקציה כלשהי של $o_1, ..., o_n$ (הפונקציה יכולה להיות, למשל, שכבות נוספות ברשת) ואז

$$\frac{dL}{dU} = \sum_{i=1}^{n} \frac{dL}{do_i} \cdot \frac{do_i}{dU}$$

כלומר, על־אף מורכבות הביטוי, אנו יודעים לגזור אותו ולחשב את כלל השרשרת, ממש כמו ברשתות fully connected. לכן האימון של רשתות שכאלו יתבצע באותו האופן, עם תהליך back propagation רגיל.

1.2.2 סיבוך של המודל

שימו לב כי איננו מוכרחים לחזות פלט מכל שלב ברשת. נוכל להתבונן בארכיטקטורה הבאה:



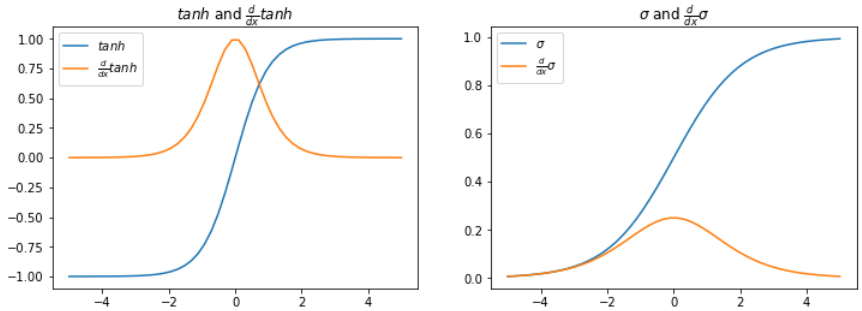
איור 2:

כאן אנו חוזים את הפלט רק בסוף המעבר על כל הקלטים, והשינוי לארגיטקטורה הוא מועט מאוד. הארכיטקטורה הזו גמישה מאוד: נוכל למשל לוותר גם על חלק מהקלטים לרשת, במקרה של חוסרים במידע.
הדבר מאפשר לנו לבנות מודל של סדרה לסדרה, sequence to sequence, כפי שראינו בהתחלה, אך גם מודל של סדרה לפלט יחיד (או לכמה פלטים, שכולם יוצאים מתוך ה־hidden האחרון). בדוגמה שלנו, של חיזוי המילה הבאה במשפט, יתכן שנעדיף מודל שכזה. למעשה, ניתן לבחור להוציא "פלטים" מחלק מה־hidden, אך לא מכולם. בבעיות רבות אין לכך משמעות, שכן מהותית אין כמה פלטים לבעיה. בבעיות בהן יש לכך משמעות נעדיף להשתמש בכך, כפי שנסביר מיד, בחלק על vanishing gradients.
המאפיין היחידי שמוכרח להישמר בארכיטקטורת RNN הוא ה־hidden, שמבטא, כאמור, את ה"זכרון" של השכבה.
ראינו כי s_{t+1} מחושב בנוסחה רקורסיבית באמצעות s_t. אולם מהו s_0? ניתן לאתחל אותו לערך שרירותי, אך נהוג להתייחס אליו כאל משקולות נלמדות ברשת: נוכל להחליף אותו בוקטור של משקולות, לגזור לפיו את ה־loss ולעדכן גם אותו בתהליך האימון, כך שנגיע לאתחול "אופטימלי" עבור הרשת שלנו.

1.3 Vanishing gradients

ראינו בתרגיל כי ברשתות עמוקות יש בעיה של vanishing/exploding gradiennts. הדבר נובע מהפעלה חוזרת של פונקציות אקטיבציה, כאשר בתהליך ה־back propagation הדבר מתרגם לכפל באיברים קטנים מ־1.

איור 3:

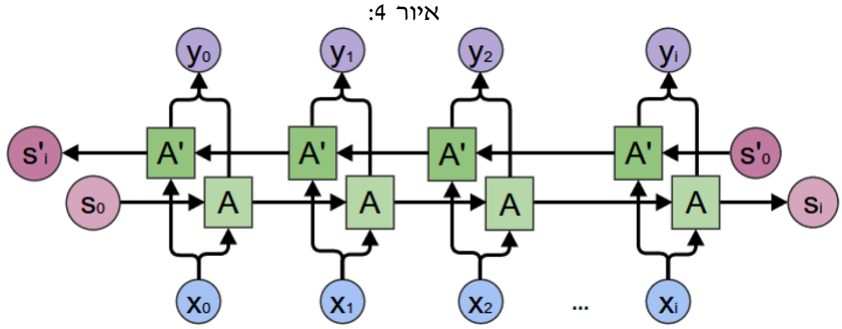


במודל RNN נגרמת תופעה דומה: עדכונים הקשורים לקלט הראשון בסדרה מגיעים "מוחלשים", כיוון שהגרדיאנטים הקשורים אליהם, הרכיבים הקשורים בנגזרת לפי ה־hidden states הראשוני, דועכים. אפקטיבית אנו יכולים לקבל את אותו האפקט שהיינו מקבלים עבור רשת נוירונים fully connected בת עשרות שכבות. במודל ה־RNN שהגדרנו הקלטים המאוחרים בסדרה משפיעים יותר על תהליך האימון, שכן הגרדיאנטים המתקבלים מהם לא דועכים באותה המידה. זהו אפקט שלא התכוונו אליו בבנייה המקורית, ונרצה לנטרל אותו. מובן כי השיטה הקלה ביותר לשם כך היא למזער את האורך של סדרות איתן נעבוד. זהו איננו פתרון לבעיה, אך במקרים רבים זהו הפתרון הקל והיעיל ביותר.
אחת הדרכים להתמודד עם הבעיה היא באמצעות שיטת gradient clipping. בשיטה זו אנו אוכפים על הגרדיאנטים להיות בעלי גודל, נורמה, בתחום מסוים. כך אנו מצליחים להעביר דרך הרשת את הכיוון אליו מצביע הגרדיאנט בצורה טובה, אך אין לנו הבטחה כי הצעד שניקח בכיוון הגרדיאנט יהיה "צעד בכיוון הנכון": אנו עשויים "לצעוד יותר מדי", או "פחות מדי". ישנה סיבה טובה לתופעת ה־vanishing gradients: הגרדיאנטים מגיעים חלשים יותר אל הקלטים הראשונים כיוון שבפונקציה שרשת הנוירונים מבטאת הם מהותית "פחות חשובים". עיוות מלאכותי של הגרדיאנטים עשוי לגרום לקשיי התכנסות.
שיטה אחרת היא, כפי שהזכרנו קודם לכן, שימוש ב"נקודות משיכה", כלומר, הוצאה של פלטים מתוך ה־RNN במקרה בו אנו מחזירים מהשכבה וקטור יחיד. הדבר יאפשר "לגזור loss" עבור חלקים מוקדמים יותר בקלט, ולקבל גרדיאנטים הקשורים לחלקים אלו ברשת באופן פחות אמצעי. כלומר, החלקים הקדומים ברשת ישפיעו עם פחות "תיווך" של סדרת קלטים ו־hidden states ארוכה, כך שההשפעה הפוטנציאלית שלהם תגדל.
נראה כעת עוד כמה מודלים ברמות סיבוך משתנות שמטרתם להתמודד עם התופעה.

1.4 Bi-Directional

ראינו כי הקלטים המוקדמים משפיעים מעט על תחזית הרשת, בעוד הקלטים המאוחרים משמעותיים יותר. אם נרצה שהקלטים המוקדמים ישפיעו יותר, כל שעלינו לעשות הוא להפוך את סדר הסדרה, ולקבל RNN "הפוך". כיצד נשמור על חשיבות האיברים האחרונים בסדרה? ובכן, נוכל להפעיל את ה־RNN המקורי יחד עם זה ההפוך ולחבר את התוצאות שלהם. החיבור יבוצע באמצעות שרשור וקטורי הפלט. נקבל RNN דו־כיווני, Bi-Directional, כבשרטוט:

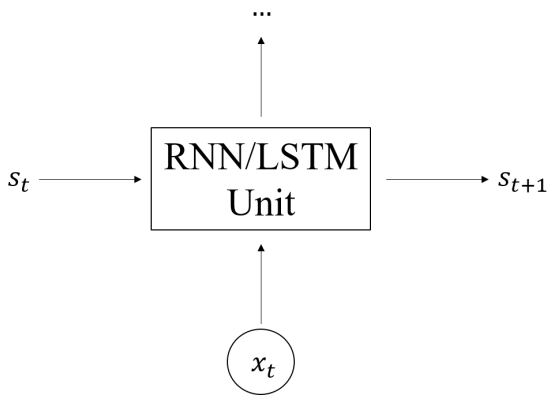
איור 4:



שני ה־RNN־ים חיים במקביל ופועלים על הסדרה ללא קשר ביניהם, עד אשר מתבצע חיבור בין הפלטים שלהם. אם כל אחד מהם מחזיר פלט יחיד, החיבור הוא שרשור הפלטים. בחיזוי sequence to sequence החיבור יעשה לזוגות של פלטים, כאשר נהוג לחבר את הפלט ה־i עם הפלט ה־$n-i$, כבשרטוט.
ארכיטקטורה שכזו מאפשרת לתת חשיבות גם לקלטים קדומים. מובן כי היא איננה פתרון מוחלט, וכי עבור מחרוזות ארוכות נסבול מגרדיאנטים חלשים גם במרכז הסדרה. פתרון זה מאפשר "לצמצם את מימדי הבעיה", במחיר נמוך, של הוספת כמות משקולות של שכבה נוספת.

1.5 LSTM (Long Short Term Memory, 1997)

נציג כעת פתרון אחר לבעיית ה־vanishing gradients, המבוסס על ארכיטקטורה מורכבת יותר לרשת שתקבל קלט סדרתי. ראשית נכליל את מודל ה־RNN שראינו קודם לכן:



איור 5:

נתייחס אל יחידת ה־RNN כאל יחידה לוגית. פעולתה, כזכור, הייתה

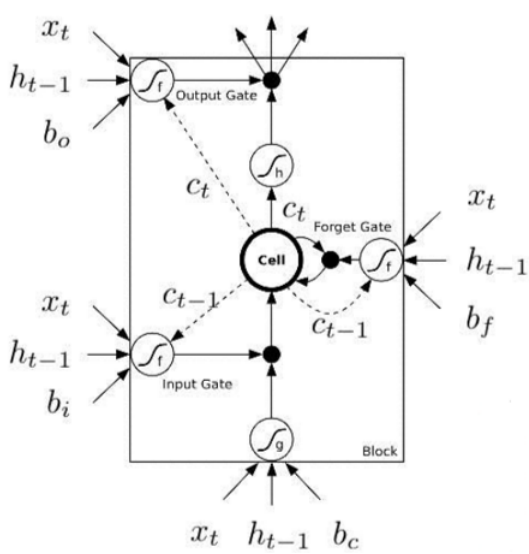
$$s_t := \sigma(U \cdot x_t + W \cdot s_{t-1})$$
$$o_t := V \cdot s_t$$

ב־LSTM גם־כן נרצה יחידה לוגית שמקבל את s_t ואת x_t ומחזירה s_{t+1} ו־o_t, אלא שפעולתה תהיה מסובכת יותר. כלומר, החלק שמיד "נסבך" הוא היחידה הלוגית, ולא אף רכיב אחר בארכיטקטורה.
ב־LSTN נרצה מנגנון שיאפשר את התכונות הבאות:

- לשכוח את הידע הקודם
- להתחשב בקלט חדש
- לתת "הגבר" כלשהו לפלט

לכל מטרה שכזו נייחד ביטוי, כמו ה־hidden s_t, אליו תתלוונה מטריצות משקולות, ה"מתווכות" הקשורות אליו. נחשוב על הביטוי כעל חלק ביחידה הלוגית שיכול לבטא את המשמעות שנייחס לו. נגדיר את הגדלים הבאים:

$$\begin{cases} input: & x_t \\ forget\,gate: & f_t = \sigma_g(W_f \cdot x_t + U_f \cdot s_{t-1} + b_f) \\ input\,gate: & i_t = \sigma_g(W_i \cdot x_t + U_i \cdot s_{t-1} + b_i) \\ output\,gate & o_t = \sigma_g(W_o \cdot x_t + U_o \cdot s_{t-1} + b_o) \\ cell\,state: & c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma_c(W_c \cdot x_t + U_c \cdot s_{t-1} + b_c) \\ hidden\,state: & s_t = o_t \cdot \sigma_h(c_t) \end{cases}$$



איור 6:

נעבור בזהירות על הביטויים. את x_t, s_t אנו מכירים, ויהיה להם אותו התפקיד שהיה להם במודל ה־RNN הפשוט. ראשית נשים לב לדמיון בין הביטויים f_t, i_t, o_t והרכיב השני ב־c_t. בכולם אנו לוקחים את x_t בתיווך של מטריצת משקולות, את s_{t-1} בתיווך של מטריצת משקולות, ומוסיפים להם bias, משקולות נוספות. נחשוב על הביטוי

$$\sigma(W \cdot x_t + U \cdot s_{t-1} + b)$$

כעל דרך לקבל את המידע לתוך היחידה הלוגית שלנו. σ היא אקטיבציה כלשהי. כלומר, אנו מקבלים את המידע מ־x_t, s_{t-1} באמצעות תיווך שלהם במשקולות, חיבורם, הוספת bias והפעלת אקטיבציה.
כעת, ל־f_t נקרא ה־forget gate, ותפקידו יהיה להחליט באיזו מידה עלינו לשכוח את המידע הקודם. i_t הוא ה־input gate ותפקידו להחליט באיזו מידה ישפיע הקלט הנוכחי. ההשפעה של שניהם באה לידי ביטוי בחישוב של c_t: זו מעין גרסא "פרימיטיבית" יותר של ה־hidden של הרשת. אכן, שימו לב כי s_t הוא פונקציה כלשהי של c_t.
c_t הוא סכום משוקלל של c_{t-1} ושל ה"קלט" שלנו, כלומר הקלט של היחידה הלוגית, או הדרך בה אנו מתווכים אותו לתוך היחידה הלוגית. f_t קובע, בסכום המשוקלל, כמה יהיה משמעותי c_{t-1}. לכן אנו יכולים לפרש אותו כ"שער שכחה", המחליט באיזו מידה להתחשב בעבר. i_t קבוע כמה יהיה משמעותי הקלט החדש. האקטיבציה בשניהם היא סיגמואיד, שהם מקבלים ערכים בתחום $[0,1]$, וקובעים כמה משמעות תהיה ל־c_{t-1} והקלט החדש בחישוב ה־"hidden", או הגרסא ה"פרימיטיבית" של ה־hidden, c_t.
כעת, o_t הוא ה־output gate, והוא מתווך לנו את c_t לתוך s_t. כלומר, הוא מעין הגבר של ה־c_t כשאנו הופכים אותו ל־s_t. s_t הוא הפעלת אקטיבציה לא ליניארית על c_t, בתוספת מידה כלשהי של הגבר, o_t.
בסך הכל אנו מקבלים ליחידת ה־LSTM את הקלטים x_t, s_{t-1} ומחזירים את הפלט s_t. בסימון בחלק הקודם החזרנו גם פלט: נוכל לייצר מתוך s_tפלט חדש, או להתייחס אל s_t כאל הפלט של הרשת ואל c_t כאל ה־hidden. את השרטוט נקרא כשם שקראנו את השרטוט ל־RNN פשוט: נעבור בזהירות על החצים, ונבין אילו קלטים כל חישוב מקבל.
באיזה אופן LSTM עדיף על RNN פשוט? ראשית, בגזירה של הגרדיאנטים, נוכל לקבל גרדיאנטים הקשורים בקלט הראשון, מבלי שפעלו עליהם אקטיבציות רבות. דרך ה"ערוץ"

$$c_t = f_t \cdot c_{t-1} + ...$$

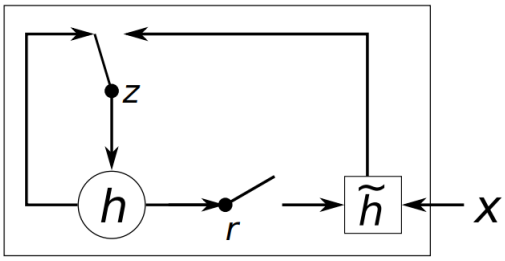
נוכל לקבל מעבר של קלט דרך הרשת מבלי להפעיל עליו אקטיבציות. לכן ישנה האפשרות שגרדיאנטים אכן יפעפעו לאחור מבלי להיחלש, ,או במילים אחרות, ישנה אפשרות להשפעה משמעותית של תחילת הסדרה על הפלט הסופי. בנוסף בסיבוך "קטן" יחסית, אנו יכולים לקבל מבנה פונקציה הרבה יותר מורכב. לכן יחידת LSTM היא באופן כללי יותר "חזקה". המחיר שאנו משלמים בכמות משקולות גדולה יותר אינו רב: אמנם הכפלנו במעט את כמות המשקולות, אך אין הדבר דומה לכלל לחלופה, בדמות רשת fully connected.
ארכיטקטורת LSTM נחשבת כיום לאחת החלופות הטובות ביותר לעיבוד של מידע סדרתי, וכפי שראינו, מהותית היא אינה שונה מארכיטקטורת RNN קלאסית: היא מערבת מעבר של hidden state, ה"זיכרון" של הרשת, ויחידה לוגית שמבצעת עליו עיבוד.

1.6 GRU (Gated Recurrent Unit, 2014)

לסיכום נראה ארכיטקטורה דומה ל־LSTM, ש"משיגה" יעדים דומים ומתחרה בה בהישגיה, אך היא מעט דלה יותר בפרמטרים. זו ארכיטקטורה חדשה יחסית (ויש שיגידו גם יותר

אופנתית). נתבונן בהגדרה שלה ובשרטוט המתאים:

$$\begin{cases} input: & x_t \\ update\,gate: & z_t = \sigma_g(W_z \cdot x_t + U_z \cdot s_{t-1} + b_z) \\ regret\,gate: & r_t = \sigma_g(W_r \cdot x_t + U_r \cdot s_{t-1} + b_r) \\ hidden\,state: & s_t = z_t \cdot s_{t-1} + (1 - z_t) \cdot \sigma_s(W_s \cdot x_t + U_s \cdot (r_t \cdot s_{t-1}) + b_s) \end{cases}$$



איור 7:

המבנה ב־GRU דומה לזה שב־LSTM. גם כאן נחשוב על ה"קלט" של היחידה במונחים של

$$W \cdot x_t + U \cdot s_{t-1} + b$$

בארכיטקטורה זו אנו מגדירים רק שני שערים. שער ה־update, z_t, מגדיר באיזו מידה נתייחס ל־hidden הקודם ובאיזו מידה נתייחס לביטוי החדש שמגיע לתוך ה־hidden. נשים לב כי כאן "ויתרנו" על אחד השערים ע"י כך שלקחנו את z_t ואת $1 - z_t$. הרכיב החדש שנכנס לתוך ה־hidden דומה ל"קלט", אלא שכאן אנו מאפשרים "לוותר" על הרכיב של s_{t-1} בתוך הקלט החדש, באמצעות שער ה־regret.
ארכיטקטורה זו, בדומה לזו של ה־LSTM, מאפשרת מעבר גרדיאנטים ללא אקטיבציות דרך הקישור

$$s_t = z_t \cdot s_{t-1} + ...$$

כך שגם כאן נוכל לקבל השפעה משמעותית של קלטים מתחילת הסדרה על הפלט.
באופן כללי ארכיטקטורות RNN הן רבות ומגוונות, ופעמים רבות ממציאים ארכיטקטורה ייעודית למטרות ספציפיות. ה־GRU, באופן מיוחד, השתרשה בקרב חוקרים ואנשי data science בזכות היותה גמישה ודלה בפרמטרים (וחלקית בזכות השמות שמעטרים את המאמר שהציג אותה)