6. Recurrent Neural Networks

היתרון של שכבות קונבולוציה על פני FC הוא ניצול הקשר המרחבי שיש בין איברים שונים בדאטא, כמו למשל פיקסלים בתמונה. יש סוגי דאטא בהם האיברים השונים יוצרים סדרה שיש לסדר האיברים חשיבות, כמו למשל טקסט, גלי קול, רצף DNA ועוד. כמובן שדאטא מהסוג הזה דורש מודל הנותן חשיבות לסדר של האיברים, מה שלא קיים ברשתות קונבולוציה. בנוסף, הרבה פעמים המימד של הקלט לא ידוע או משתנה, כמו למשל אורך של משפט, וגם לכך יש לתת את הדעת. כדי להתמודד עם אתגרים אלו יש לבנות ארכיטקטורה שמקבלת מספר לא ידוע של וקטורים ומוציאה וקטור יחיד, כאשר הוקטור היחיד מכיל בתוכו קשרים על הדאטא המקורי שנכנס אליו. את וקטור המוצא ניתן להעביר בשכבת FC או במסווג, תלוי באופי המשימה.

6.1 Sequence Models

6.1.1 Recurrent Neural Networks

רשתות רקורסיביות הן הכללה של רשתות נוירונים עמוקות, כאשר יש להן רכיב זיכרון פנימי שמאפשר לתת משמעות לסדר של איברי הכניסה. כל איבר שנכנס משוקלל ביחס לפונקציה קבועה בתוספת רכיב משתנה שתלוי בערכי העבר. כאשר נכנס וקטור x, הוא מוכפל במשקל w_{xh} ונכנס לרכיב זיכרון h_t, כאשר h_t הוא פונקציה של x_t, h_{t-1}:

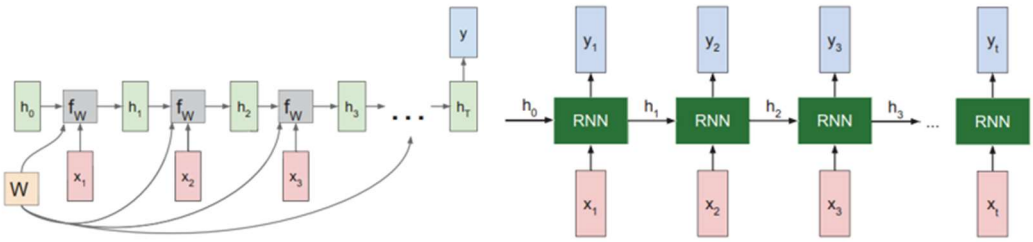
$$h_t = f_w(h_{t-1}, x_t)$$

מלבד המשקלים הפועלים על וקטור הכניסה, יש גם משקלים שפועלים על רכיב הזיכרון – w_{hh}, ומשקלים הפועלים על המוצא של רכיב זה – w_{hy}. המשקלים w_{hx}, w_{hh}, w_{hy} זהים לכל השלבים, והם מתעדכנים ביחד. כמו כן, הפונקציה f_w היא קבועה לכל האיברים, למשל tanh, sigmoid או ReLU. באופן פורמלי התהליך נראה כך:

$$h_t = f_W(w_{hh}h_{t-1} + w_{xh}x_t), f_w = tanh/ReLU/sigmoid$$

$$y_t = w_{hy}h_t$$

באופן סכמתי התהליך נראה כך:



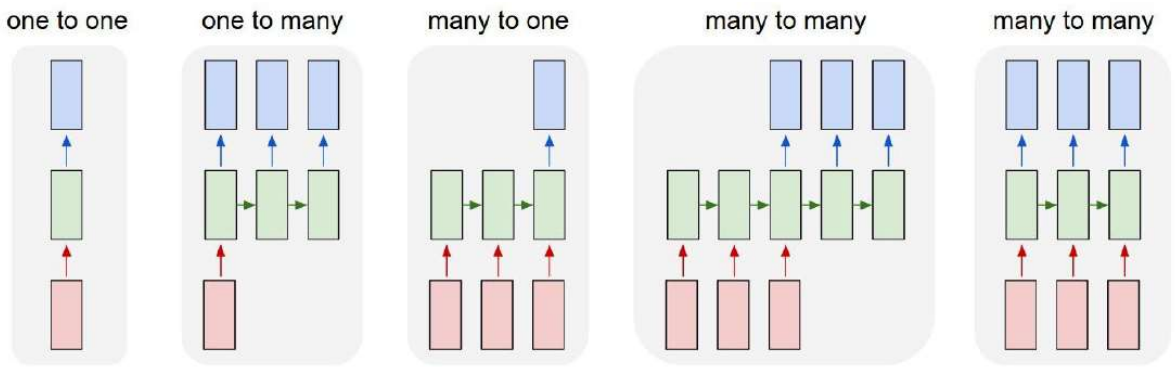
איור 6.1 ארכיטקטורות RNN בסיסיות: Many to Many (מימין) ו-Many to One (משמאל). על כל חץ יש משקל מתאים עליו מתבצעת הלמידה.

כמובן שניתן גם לשרשר שכבות חבויות ולקבל רשת עמוקה, כאשר פלט של שכבה מסוימת הופך להיות הקלט של השכבה הבאה. ישנם מודלים שונים של RNN, המתאימים לבעיות שונות:

One to many – יש קלט יחיד ורוצים להוציא מספר פלטים, למשל מכניסים תמונה לרשת ורוצים משפט שיתאר אותה (Image captioning).

Many to one – רוצים לקבוע משהו יחיד עבור קלט מרובה, למשל מקבלים משפט ורוצים לדעת מה הסנטימנט שלו – חיובי או שלילי.

Many to many – עבור כל סדרת קלט יש סדרת פלט, למשל תרגום בין שפות – מקבלים משפט ומוציאים משפט.



איור 6.2 מודלים שונים של RNN.

6.1.2 Learning Parameters

הלמידה של הרשת נעשית בצורה דומה לרשתות שבפרקים הקודמים. עבור דאטא $x = (x_1, \dots, x_n), (y_1, \dots, y_n)$ נגדיר את פונקציית המחיר:

$$L(\theta) = \frac{1}{n}\sum_i L(\hat{y}_i, y_i, \theta)$$

כאשר הפונקציה $L(\hat{y}_i, y_i, \theta)$ תותאם למשימה – עבור משימת סיווג נשתמש ב-cross entropy ועבור בעיות רגרסיה נשתמש בקריטריון MSE. האימון יתבצע בעזרת GD, אך לא ניתן להשתמש ב-backpropagation הרגיל כיוון שכל משקל מופיע מספר פעמים – למשל w_{hx} פועל על כל הכניסות ו-w_{hh} פועל על כל רכיבי הזיכרון. כדי לבצע את עדכון המשקלים משתמשים ב-backpropagation through time (BPTT) – מסתכלים על הרשת הנפרשת כרשת אחת גדולה, מחשבים את הגרדיאנט עבור כל משקל, ואז סוכמים או ממצעים את כל הגרדיאנטים. אם הדאטא בכניסה הוא בגודל n, כלומר יש n דגימות בזמן, אז יש n רכיבי זיכרון, ו-$n-1$ משקלים w_{hh}. לכן הגרדיאנט המשוקלל יהיה:

$$\frac{\partial L}{\partial w_{hh}} = \sum_{n-1} \frac{\partial L}{\partial w_{hh}(t)} \quad or \quad \frac{\partial L}{\partial w_{hh}} = \frac{1}{n-1}\sum_{n-1} \frac{\partial L}{\partial w_{hh}(t)}$$

כיוון שהמשקלים זהים לאורך כל הרשת, $w_{hh}(t) = w_{hh}$ והשינוי בזמן יהיה רק לאחר ביצוע ה-BPTT ויהיה רלוונטי רק לוקטור הבא.

הצורה הפשוטה של ה-BPTT יוצרת בעיה עם הגרדיאנט. נניח שרכיב הזיכרון מיוצג בעזרת הפונקציה הבאה:

$$h_t = f(z_t) = f(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$$

לפי כלל השרשרת:

$$\frac{\partial h_n}{\partial x_1} = \frac{\partial h_n}{\partial h_{n-1}} \times \frac{\partial h_{n-1}}{\partial h_{n-2}} \times \dots \times \frac{\partial h_2}{\partial h_1} \times \frac{\partial h_1}{\partial x_1}$$

כיוון ש-w_{hh} קבוע ביחס לזמן עבור וקטור כניסה יחיד, מתקבל:

$$\frac{\partial h_t}{\partial h_{t-1}} = f'(z_t) \cdot w_{hh}$$

אם נציב את זה בכלל השרשרת, נקבל שעבור חישוב הנגזרת $\frac{\partial h_n}{\partial x_1}$ מכפילים $n-1$ פעמים ב-w_{hh}. לכן אם מתקיים $|w_{hh}| > 1$ אז הגרדיאנט יתבדר, ואם $|w_{hh}| < 1$ הגרדיאנט יתאפס. בעיה זו, של התבדרות או התאפסות הגרדיאנט, יכולה להיות גם ברשתות אחרות, אבל בגלל המבנה של RNN והלינאריות של ה-BPTT ברשתות רקורסיביות זה קורה כמעט תמיד.

עבור הבעיה של התבדרות הגרדיאנט ניתן לבצע clipping – אם הגרדיאנט גדול מקבוע מסוים, מנרמלים אותו:

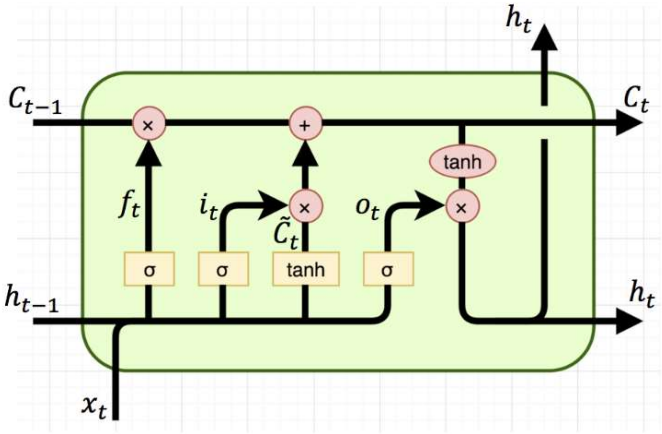
$$if\ \|g\| > c, then\ g = \frac{cg}{\|g\|}$$

הבעיה של התאפסות הגרדיאנט אמנם לא גורמת לחישובים של מספרים עצומים, אך היא בעצם מבטלת את ההשפעה של איברים שנמצאים רחוק אחד מהשני. אם למשל יש משפט ארוך, אז במקרה בו הגרדיאנט דועך במהלך ה-BPTT, כמעט ואין השפעה של המילה הראשונה על המילה האחרונה. במילים אחרות – התאפסות הגרדיאנט גוררת בעיה של Long-term, כלומר קשה ללמוד דאטא בעל תלות בטווח ארוך, כמו משפט ארוך או תופעות שמשתנות לאט. בגלל הבעיה הזו לא משתמשים ב-RNN הקלאסי, אלא מבצעים עליו שיפורים, כפי שיוסבר בפרק הבא.

6.2 RNN Architectures

6.2.1 Long Short-Term Memory (LSTM)

כדי להתגבר על בעיית דעיכת הגרדיאנט המונעת מהרשת להשתמש בזיכרון ארוך טווח, ניתן להוסיף אלמנטים לרכיב הזיכרון כך שהוא לא יכיל רק מידע על העבר, אלא יהיה גם בעל שליטה על איך להשתמש במידע. ב-RNN הפשוט לרכיב הזיכרון יש שתי כניסות – h_{t-1}, x_t, ובעזרתן מחשבים את המוצא על ידי שימוש בפונקציה $f_w(h_{t-1}, x_t)$. למעשה רכיב הזיכרון הוא קבוע והלמידה מתבצעת רק במשקלים. ב-LSTM יש שני שינויים עיקריים – מלבד הכניסות הרגילות יש עוד כניסה c_{t-1}, ובנוסף לכך h_t מחושב בצורה מורכבת יותר. באופן הזה המשקלים דואגים לזיכרון ארוך טווח של דברים, והמבנה הפנימי של רכיב הזיכון אחראי על הזיכרון של הטווח הקצר. נתבונן בארכיטקטורה של תא הזיכרון:



איור 6.3 תא זיכרון ברשת LSTM.

הצמד $[x_t, h_{t-1}]$ נכנס לתא ומוכפל במשקל w, ולאחר מכן עובר בנפרד דרך ארבעה שערים (יש לשים לב שלא מבצעים פעולה בין x_t ל-h_{t-1} אלא הם נשארים בנפרד ואת כל הפעולות עושים על כל איבר בנפרד). השער הראשון $f_t = [\sigma(x_t), \sigma(h_{t-1})]$ הוא שער שכחה והוא אחראי על מחיקת חלק מהזיכרון. השער השני i_t הוא שער כניסה והוא אחראי על כמה להתייחס למידע החדש. השער הרביעי o_t הוא שער מוצא והוא אחראי על כמה מהזיכרון רלוונטי לדאטא הנוכחי שנכנס x_t. שלושת השערים האלו נקראים מסכות (Masks), והם מקבלים ערכים בין 0 ל-1 כיוון שהם עוברים דרך סיגמואיד. יש שער נוסף $\tilde{c}_t$ (לפעמים מסומן באות g) שאחראי על השאלה כמה לכתוב לתא הזיכרון. באופן הזה מקבלים במוצא לא רק את h_t אלא גם את c_t.

ארכיטקטורת הרכיב מאפשרת להתייחס לאלמנטים נוספים הקשורים לזיכרון – ניתן לשכוח חלקים לא רלוונטיים של התא הקודם (f_t), להתייחס באופן סלקטיבי לכניסה (i_t) ולהוציא רק חלק מהמידע המשוקלל הקיים (o_t). באופן פורמלי ניתן לנסח את פעולת התא כך:

$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot \tilde{c}, h_t = o \odot \tanh(c_t)$$

כאשר האופרטור $\odot$ מסמל כפל איבר איבר (כיוון שלשערים נכנס הזוג $[x_t, h_{t-1}]$, יש לבצע מכפלה בכל אחד מהאיברים).